Explainable Machine Learning Models: Review and Computational Analysis

Šimon Růžička, Utku Ipek
Technical University of Munich
{simon.ruzicka, utku.ipek}@tum.de

Abstract

Many innovative models providing explained decisions have appeared in the past years as a part of the quickly growing field of explainable machine learning. Each model fits different use cases, based on the task, input type, computational capabilities, and priorities in the accuracy-explainability trade-off. We provide a brief overview of multiple explanation approaches and describe their differences with special regard for their computational aspects. These include the time and memory complexity of the algorithms, as well as any possible optimizations. We highlight the difference between post-hoc and transparent explaining models and their different use cases. We focus in detail on three general groups of explaining models – simple model-agnostic explainers, models explaining predictions on images using focus maps, and inherently transparent predicting models. We provide multiple examples of models from each of the mentioned groups, describing their algorithms and comparing them against each other.

Index Terms

machine learning, explainability, XAI, accuracy-explainability trade-off, post-hoc explanations, focus mapping, transparent models.

I. Introduction

COMPLEX machine learning models are becoming more and more accessible in the past years, finding their use cases in new fields of industrial and consumer products. With more accessible hardware, the model complexity often grows, usually with the only goal of a higher predicting accuracy. However, calls for another approach to machine learning have appeared, asking for an intuitive human understanding of the model and its decisions, rather than only more accurate predictions.

There are various different terms for this phenomenon, often used synonymously, which we will try to distinguish. Explainable machine learning[1] is the most commonly used term, usually generalizing this whole field of scientific knowledge. However, as was described by Gilpin *et al.* [1], there are multiple aspects of an explanation, not all of which are always satisfied. Authors distinguish between *interpretability*, being simple enough to be understandable by humans, and *completeness*, describing the operations of the model accurately, as the two main aspects of an explanation. The authors put emphasis on a balance between these two aspects. An explanation that is highly interpretable might be popular among the users, but it can be lacking in its completeness as it may simplify the problem way too much, which is certainly a cause for concern. This is also the main problem with human evaluation of explaining systems – the aspect of explanation completeness is hard to judge without knowing the model internals.

This decomposition of explainability aspects implies that our definition of the term *explainability* contains more information about a model than *interpretability*, adding the completeness aspect that the produced explanations represent the real decision process accurately. However, the usage of these words does not necessarily reflect this fact. In scientific communities, the latter term is used more commonly, even for systems that are supposed to be complete in their reasonings. This was shown by Adadi & Berrada [2] when analyzing term popularity in search engine trends. However, in the common public, the term explainability prevailed. Due to the rapid growth of this scientific field, there is no generally accepted taxonomy; meaning that the specific properties of a model should always be checked more thoroughly.

A. Accuracy-explainability trade-off

One of the many problems faced by the explainable AI researchers is the effect that their explanation-producing models have on the prediction itself. The predicting accuracy of a black box model is usually tied to its complexity – which often causes problems to the tied explaining model, making it either less interpretable or complete. This problem means that there is a clear trade-off between model accuracy and our ability to explain its reasoning [3]. The predicting and explaining models, alternatively just one transparent model producing both, must be chosen adequately for the needed task.

In low-risk scenarios such as recommender systems, we might prioritize good predictions over accurate explanations. In these cases, *post-hoc* explaining models can be added to existing deep learning models. As the name suggests, these models produce an explanation after the prediction has been made, meaning that they do not affect the prediction accuracy as long as

[1]Even though the difference between machine learning and artificial intelligence is significant, this term is commonly exchanged with explainable AI or abbreviated as XAI; possibly due to the fact that the XML abbreviation already has another meaning in this domain (Extensible Markup Language).

we keep the underlying model the same. For high-risk scenarios such as medicine or criminal justice, prediction is generally considered worthless without good reasoning, as the final decision is made based on a consensus of multiple qualified people. Simpler *transparent* models are often used in such cases. As the explanations are implicit within the model, they are complete by definition, which is highly desirable. However, provided explanations such as weight coefficients of a linear model usually have to be transformed to improve their interpretability. In our work, we will focus on both model categories by showing multiple specific examples in detail.

Another advantage of transparent models is that they produce *global* explanations, meaning they interpret the entire model, and are therefore applicable on all made predictions. On the other hand, *local* explanations are only valid for a single prediction and are not applicable globally. This distinction is important when we want to show that our model does not discriminate based on undesired features, for example race or gender; in such cases, global explanation of a model may be desirable or even necessary.

II. General explanations

As briefly introduced before, the need for interpretability has surged in recent years, driven by the increased accuracy of the black-box models, made possible by the recent advancements in the field of AI. For us to trust the AI systems, the decision-making process of these systems must become transparent. So, the specific decisions made by those systems should be interpretable, especially in the areas impacting human lives directly, like healthcare. In this chapter, several methods will be presented, sharing the same purpose according to the taxonomy map generated by Linardatos *et al.* [3], which is to explain black-box/complex models.

The chapter is named General Explanations since the explanation methods summarized here are model-agnostic. This implies their applicability across diverse models, ensuring flexibility and broad utility. Moreover, the techniques discussed here are not restricted to images, meaning they mostly handle various data types like images, text, and tabular. Before diving into the methods, it is also important to note that they serve a dual purpose in scope. They can offer local explanations, that is, the explanation behind predictions for a single instance of data, or global explanations, delivering an explanation for the decision-making process of the entire model.

A. Local Interpretable Model-agnostic Explanations (LIME)

As Gilpin *et al.* [1] stated, the idea behind linear proxy models lies in the complex models, which have many learnable parameters and many operations to execute and are hard to interpret. To ease the interpretation process, the complexity of those models can be reduced. A new method called a proxy model can be proposed, exhibiting behavior analogous to the original model. The Local Interpretable Model-agnostic Explanations (LIME) [4] proxy model has been summarized in this part. It is one of the most renowned interpretability methods.

The method has been proposed by Ribeiro *et al.* [4] to address two main problems: "trusting a single prediction of a model" and "trusting the whole model." While the first is related to local explanations, the latter is related to global explanations. At its core, LIME is a method that provides local explanations, and its inner workings offer a solution to the first problem. Although the method is for local explanations, they upgraded it by generating a subset of explainable instances to solve the second problem. Before diving into the technical details of LIME, it should be mentioned that there are several characteristics described by them.

- Being Interpretable (see section I-A): They state that it is subject to the audience, and the features of the explanation model may not be the same (true) features of the original model.
- Model-agnostic: The main idea behind LIME is flexibility, meaning any classifier existing should be able to be explained.
- Local fidelity: They state that the model should have local fidelity, meaning that for an explanation to make sense, it should at least function as expected around a single instance during the prediction process. An important point here is that achieving local fidelity does not mean achieving global fidelity.

To describe an explanation, Ribeiro *et al.* [4] represent it as an interpretable model. This model belongs to a class including all potentially interpretable models. An essential common feature of these models is that as their domain, they have a vector of interpretable components, having binary values based on the existence/non-existence of those components. In addition to an interpretable model, they added a complexity term to the formula, which measures the complexity of the corresponding interpretable model. Then, they defined a locality-aware loss function, taking the original model with no assumptions since LIME is model agnostic, the interpretable model, and a proximity measure as arguments. This proximity measure describes the locality around the instance being predicted. Using the proximity measure, instances can be sampled around the interpretable representation of the primary instance. The loss function can be minimized, and the explanation can be obtained.

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{1}$$

$$\mathcal{L}(f, g, \pi_x) = \sum_{z, z' \in \mathbb{Z}} \pi_x(z)(f(z) - g(z'))^2 \tag{2}$$

$$\pi_x(z) = \exp\left(-\frac{D(x, z')}{\sigma^2}\right) \tag{3}$$

Above, the equations presented by Ribeiro *et al.* [4] can be observed. Equation 1 shows the explanation generated by the method, where G constitutes the group of models with potential interpretability, $\Omega(g)$ quantifies the complexity level of explanation g, f denotes the model under explanation, and $\pi_x(z)$ 3 serves as a measure quantifying the proximity between the instances z and x. In equation 3, D stands for some distant function, and the selection of it depends on the problem. An important note is that z and x denote the initial representations of the instances. The interpretable forms are represented by binary vectors z' and x'. Finally, Equation 2 shows the loss function to be minimized.

B. Learning disentangled representations of data

The taxonomy Gilpin *et al.* [1] created places disentangled representations below the explanation-producing systems. They listed different approaches to learning them like variational autoencoding [5], InfoGAN [6], beta-VAE [7], etc. Data representations can be learned through various models, which shows that it is not a model-specific method. This subsection won't go into details of these approaches. Instead, the main focus will be various works on defining disentangled representations formally and evaluating them quantitatively.

In their paper, Bengio *et al.* [8] state that the choice of data representation boosts the performance of AI algorithms. That makes feature engineering a crucial aspect of developing and running these algorithms. However, relying on high-quality feature engineering each time when trying out a new method slows down the progress in AI. So, to speed up the progress, the load on feature engineering can be lightened. Instead, a model can learn the data representations to quickly identify useful features. Still, it is also important to note that the data representations learned should be adequate. To determine how good a data representation is, they defined some priors containing smoothness, multiple explanatory factors, a hierarchical organization of explanatory factors, and several more. The key takeaway about these priors can be summarized as they pave the way for disentangling the factors of variation, which refers to different features that contribute to changes in data. These kinds of features have the attribute of being explanatory, and learning these elements can be shown as an example method of explaining AI models.

Formal definition: As Higgins *et al.* [9] state, the motivation behind defining disentangled representations formally lies in there needing to be a consensus on the answers to questions like: How are the data generative factors formed, how should they be represented, etc.? No formal definition makes it hard to quantify the advancements in the field. So, the main idea in generating a definition is to use natural transformations from the real world. They state that these natural transformations are symmetry transformations and have a crucial property: being able to change some features of the world while the remaining features remain invariant. Using group theory, they entitled a vector representation disentangled regarding a specific decomposition of a symmetry group. An important point is that this decomposition shall decompose the group into dissociated subspaces, each influenced by only one of the subgroups, and remain unaffected by the actions of other subgroups.

More technically, given the group G, the world state W, and the vector space Z, Higgins *et al.* [9] state that Z is disentangled relating to the decomposition $G = G_1 \times \ldots \times G_n$ if an action $\cdot : G \times Z \to Z$ exists, a mapping $f : W \to Z$ exhibits equivariance between the transformations applied to W and Z, and there exists a decomposition $Z = Z_1 \times \ldots \times Z_n$ where each Z_i solely influenced by G_i, and remain fixed otherwise.

Quantitaive evaluation framework: Previously, several models for learning representations of data have been mentioned (see section II-B). In their paper, Eastwood *et al.* [10] state that visual inspection is the current standard method for evaluating disentangled representations learned by these models. To deal with this issue, they generated a quantitative framework to assess the representations. Preceding the exploration of this framework, a key point ought to be understood: For the framework to work, the ground-truth latent space needs to be obtainable. This leads to explicitly specifying the requirements for the disentangled representations and quantifying them to analyze the models fairly. To briefly mention the framework they created, it consists of four courses of action:

- First, the model M should be trained using a synthetic dataset featuring generative factors $\boldsymbol{z}$.
- A disentangled code $\boldsymbol{z}$ should be extracted for every data point x in the dataset ($\boldsymbol{c} = M(\boldsymbol{x})$). Note that c has the dimension D, and z has the dimension K. While the case $D > K$ means that $\boldsymbol{c}$ incorporates irrelevant dimensions, the case $D = K$ means that $\boldsymbol{c}$ can be obtained through the permutation operation over $\boldsymbol{z}$.
- A regressor should be trained to estimate z ($\hat{\boldsymbol{z}} = f(\boldsymbol{c})$). In detail, to estimate K generative factors, the K number of regressors needs to be trained. Then, a matrix R containing relative importance values can be obtained. An entry R_{ij} in the matrix exhibits $\boldsymbol{c}_i$'s importance in estimating $\boldsymbol{z}_j$.
- Finally, the regressor's divergence can be assessed by comparing it with the ideal mapping and the error in prediction.

Eastwood *et al.* [10] defined three bases for explicit and quantifiable representations: disentanglement, completeness, and informativeness. The portrayal of disentanglement's quantification can be evidenced by the formulas in 4 where D_i indicates the score of $\boldsymbol{c}_i$, $H_K(P_{i\cdot})$ indicates the entropy, and P_{ij} represents the likelihood of $\boldsymbol{c}_i$'s significance in estimating $\boldsymbol{z}_i$.

$$
\begin{aligned}
D_i &= \left(1 - \frac{H_K(P_{i.})}{K-1}\right) \\
H_K(P_{i.}) &= -\sum_{k=0}^{K-1} P_{ik} \log_K P_{ik} \\
P_{ij} &= \frac{R_{ij}}{\sum_{k=0}^{K-1} R_{ik}}
\end{aligned}
\tag{4}
$$

Completeness can be defined as the measure of how much a single code variable represents each underlying factor and calculated with the formulas given in 5. Here, $H_D(\tilde{P}_{.j})$ stands for the entropy again.

$$
\begin{aligned}
C_j &= \left(1 - H_D(\tilde{P}_{.j})\right) \\
H_D(\tilde{P}_{.j}) &= -\sum_{d=0}^{D-1} \tilde{P}_{dj} \log_D \tilde{P}_{dj}
\end{aligned}
\tag{5}
$$

Finally, informativeness can be defined as the quantity of information contained within a representation regarding the underlying variations. This criterion is quantified by the prediction error mentioned in the last step of the framework they defined. This error $E(z_j, \hat{z}_j)$ is calculated using the estimated generated factors and the ground truth ones. In their experiment with the framework, they decided to compare the representations learned by PCA [11], VAE, beta-VAE, and InfoGAN. As mentioned before, their dataset is synthetic, containing an object in various poses and colors. To evaluate them in an equal setting, they ensured that all the models had the same ResNET [12] architecture and were trained with the same number of code variables. Their results showed that while PCA has the worst scores, InfoGAN reaches the highest in disentanglement, completeness, and informativeness.

III. Explaining images with focus maps

Machine learning tasks working with images, such as image classification, image captioning or visual question answering (VQA) [13] are commonly chosen for experimenting with new methods in the field of XAI. One possible reason might be that these models have significantly higher input sizes when compared to models working on tabular data or text, which usually results in increased model complexity. Using any easily interpretable transparent model for constructing any reliable prediction is not feasible in most cases; neural networks mostly solve such complex tasks.

In addition, one of the biggest powers of these explaining techniques is actually the form of the explanation itself. Having the model input in a visual form can help us form a so-called "visual explanation" in ways not available for other input types. Typically, this is done by highlighting certain parts of the input image, which the model considers critical, essentially creating a focus map[2]. However, the interpretation of focus maps and their highlighted image regions can vary:

- Visual approach – highlighted region contains more pixel value changes, meaning that the region carries more information
- Human attention approach – when testing human subjects, highlighted regions did receive most of their attention (see section V-D)
- Explanation approach – changes in highlighted regions would affect a prediction of an explained model the most

Even though these different approaches to focus map construction can sometimes strongly correlate, it is not necessarily a rule. It is always essential to keep the interpretation of a given focus map in mind when making any conclusions. While discussing explainable machine learning, we will mostly encounter the last of the approaches mentioned, as our models commonly aim to produce such a focus map. However, the remaining approaches can sometimes help us produce more accurate predictions or explanations [14] [15].

A. Gradient methods

When attempting to visually explain the inner processes of a neural network by a focus map, the simplest way to construct such a focus map would be to utilize the back-propagated gradient. In a typical unexplained neural network, the gradient is calculated only during the training process, mostly due to its relatively high computational complexity compared to the forward pass. By calculating the gradient in the learning process, we determine which model parameters/weights must change to improve the network prediction. However, we can easily reformat this algorithm to determine which inputs/pixels must change to affect the network output the most. This aligns with the explanation approach of creating a focus map we defined earlier.

Simonyan *et al.* [16] further studied this approach and described the following algorithm. The process itself seems quite straight-forward – classify the input image I_0 of dimensions $m \times n$ to get the predicted class c, then compute the derivative of the class score function S_c by the flattened input image I:

[2]Focus maps are also commonly called saliency maps or attention maps, based on context and type of literature. Unlike other sources, we will not distinguish between these sources and use only focus maps from now on.

$$\mathbf{w} = \frac{\partial S_c}{\partial I}(I_0) = \nabla S_c(I_0) \in \mathbb{R}^{mn} \tag{6}$$

Afterward, the gradient is used for constructing $\hat{S}_c$, a locally linear estimate of the class score function around I_0. We calculate the intercept $b = S_c(I_0)$ and construct first-order Taylor polynomial:

$$\hat{S}_c(I) = \mathbf{w}^T I + b \tag{7}$$

This estimate shows that a greater absolute value in gradient implies a greater change in the class score. Therefore, high-gradient regions are more likely important for correct classification. For images with multiple channels, such as RGB channels of colored images, this entire process is performed for each channel, with the resulting focus value of the pixel being the maximum of the gradient values across all channels.

In addition, adaptation of this process to object localization task was described in the article. From the resulting focus map, a thresholding operation was performed to separate foreground and background pixels; with the thresholds being set as desired quantiles of the focus map values. GraphCut algorithm [17] is then used for the image segmentation process. The largest connected component of foreground pixels is the prediction of where the localized object should be.

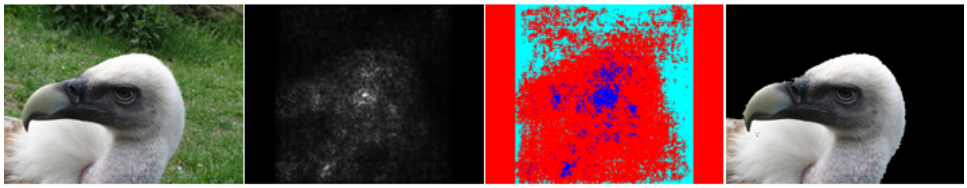

Fig. 1. One of the examples shown by Simonyan *et al.* [16]. *Left:* source image. *Left-middle:* the corresponding focus map for the top-1 predicted class. *Right-middle:* thresholded focus map: blue shows the areas used to compute the foreground color model, cyan – background color model, pixels shown in red are not used for color model estimation. *Right:* the resulting foreground segmentation mask.

Even though this method seems simple to understand and implement due to its similarity to the training process, the computational complexity is higher than for other methods producing focus maps. This problem lies in the propagation through all of the layers, commonly convolutional, towards the original image. This process can be optimized by stopping our explaining process at an earlier layer, effectively replacing the image input with the k-th convolutional layer output. However, this comes at the cost of losing the sharpness of our focus map, as further layers usually have lower resolution and some information is lost in the convolution processes.

In addition, the presented gradient methods come with another significant drawback. As we highlight areas that affect the resulting prediction the most, we ignore the important reality of how they affect it. These methods will highlight areas of an image that decrease the class score significantly similarly to those that increase it. For example, if an image we used to classify animals contained a dog and a cat, these areas would get highlighted in the focus map, regardless of the prediction, with the reasoning that changes in these regions could affect the output prediction the most. More complex focus mapping methods, described further in the text, tackle this problem by using specialized architectures.

B. Class activation maps

Another method called Class activation map, commonly abbreviated as CAM, working directly with the structure of a special convolutional neural network was brought up by Zhou *et al.* [18]. The used network consists of multiple convolutional layers $\mathcal{L}^{(1)}, \ldots, \mathcal{L}^{(k)}$, operation of global average pooling (abbreviated as GAP), and exactly one fully connected layer $\mathcal{L}^{(k+1)}$. Thanks to the pooling operation, the fully connected layer has very few inputs, corresponding to the number of channels in the last convolutional layer, which we will denote as n. Therefore, assuming we are making a prediction into C classes where the predicted class is denoted as c, the weight matrix of this layer $\mathbf{W}$ will have dimensions $n \times C$.

When attempting to explain a prediction made by the network, the GAP operation is "left out" of this process. We can utilize the channels of the last convolutional layer $\mathcal{L}_1^{(k)}, \ldots, \mathcal{L}_n^{(k)}$, each multiplied by the weight $\mathbf{W}_{i,c}^{(k+1)}$ from the fully connected layer $\mathcal{L}^{(k+1)}$; corresponding to the input channel $i \in \{1, \ldots, n\}$ and the predicted label c as the layer output. What we get is a linear combination of these channels called Class activation map M_c (equation 8), essentially showing image regions with high activation in $\mathcal{L}^{(k)}$, relative to the weight of that convolutional layer. How much a region is highlighted in CAM represents the effect of this region on the resulting prediction. This process is shown in the image 2.

$$M_c = \sum_{i=1}^{n} \mathbf{W}_{i,c} \mathcal{L}_i^{(k)} \tag{8}$$

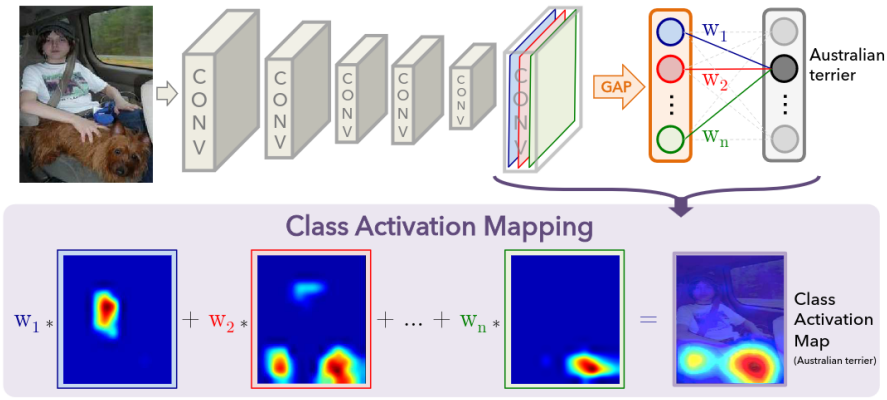

Fig. 2. Visualization of CAM created by Zhou *et al.* [18]. The predicted class c is mapped back to the previous convolutional layer $\mathcal{L}^{(k)}$ to generate the class activation maps (CAMs). The CAM highlights the class-specific discriminative regions.

Compared to the gradient method of generating a focus map, we notice that the problem of highlighting image regions suggesting other class predictions is effectively gone. Unlike gradient, the CAM is generated from the activation of layer $\mathcal{L}^{(k)}$ and the vector of weights $\mathbf{W}_{:,c}^{(k+1)}$; both of which directly serve to calculate the prediction when combined with the GAP operation. Creating the prediction and the explanation from the same sources effectively solves this problem.

An algorithm for solving the task of object localization using this method was described by Zhou *et al.* as well. Already existing CNN architectures were used for this task, with the only modification being removing all but one fully connected layer. After producing the prediction c and Class activation map for the top predicted class M_c, a 20 % thresholding operation was performed on the CAM. The bounding box of the largest connected component was chosen to predict the object's position.

From the method description, it becomes apparent that the computational complexity of this method is lower than computing the gradient for one input image. All computations occur only on the final layer, making this method surprisingly fast for deeper networks; as the previous layers are used only for classification, and additionally, the final convolutional layer is typically smaller in size, due to the nature of convolution and pooling operations. However, it is important to remember the assumption of this method when interpreting the focus map – regions of activations in the $\mathcal{L}^{(k)}$ layer should roughly correspond to regions of input image pixels. For deeper networks, this relationship gets more entangled, meaning that the explaining power of the focus map is lower. The computation of the Class activation map is then essentially a linear combination of vectors, which is highly optimized on modern GPUs with the Multiply-Accumulate operation [19]. This difference is especially notable when compared to other operations such as computing gradients of non-linear functions.

C. Grad-CAM

The biggest drawback of the conventional Class activation map technique is the limited amount of neural network architectures it can use. Any fully connected neural layer following the convolutional layers had to be removed, which affected the resulting accuracy. In addition, this method could not be used for tasks with more complex inputs or outputs, such as Visual question answering with both textual and image input or image captioning with text output instead of categorical. Therefore, Selvaraju *et al.* [20] proposed a generalizing modification of this method named Grad-CAM.

As its name suggests, Grad-CAM is in principle a gradient method, enhancing the basic CAM with this approach. The network used for classification is effectively split into two parts – The first consists of purely convolutional layers taking the image as input, and the second part will contain the neural layers of different architecture, such as fully connected, recurrent, etc., and other operations. The entire second part is considered a black box, which we use to produce decisions in the forward pass and utilize its gradient in the backward pass when producing an explanation. This description is backward compatible with basic CAM, where the "second part" was only the single fully connected layer with a GAP operation. Same as in CAM, the activation maps are taken from the last convolutional layer, denoted as A for simplicity[3].

As a result of the forward pass, class c was predicted as the most likely from the output y_c (before softmax). In the explaining backward pass, we calculate the gradient of y_c of the entire non-convolutional part of the network for k-th channel, $\frac{\partial y_c}{\partial A^k}$. This gradient has the same shape as one channel of the layer A, but global average pooling is applied to it, effectively turning it into a weight of the importance of the channel A^k for class c, denoted as $\alpha_k^c \in \mathbb{R}$ (similarly to the weight $\mathbf{W}_{k,c}$ in CAM, see equation 9). Afterward, the activations of the convolutional channels A^k are multiplied by their weights α_k^c and summed up.

[3]Consistently with previous section III-B, this layer would be called $\mathcal{L}^{(k)}$. However, keeping the notation simple and consistent with the original paper is in our best interest.

The final focus map L^c is then created by applying the ReLU function on the individual numbers in this linear combination (see equation 10). Visualization of this process can be seen in the figure 3.

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y_c}{\partial A_{i,j}^k} \tag{9}$$

$$L_{i,j}^c = \text{ReLU}\left(\sum_k \alpha_k^c A_{i,j}^k \right) \tag{10}$$

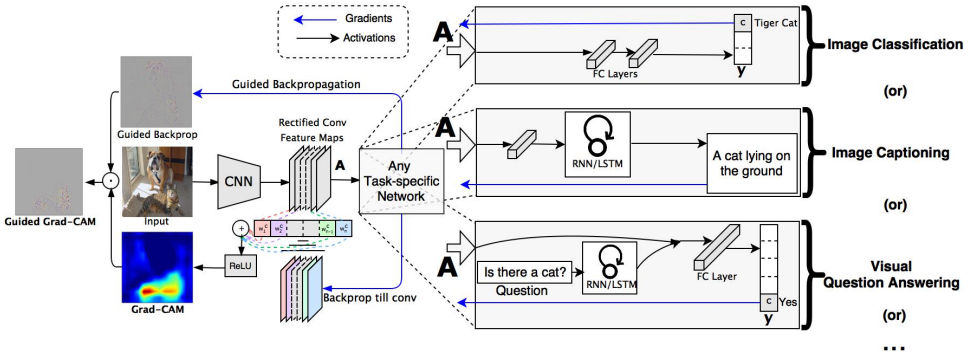

Fig. 3. Visualization of Grad-CAM created by Selvaraju *et al.* [20]. As opposed to CAM, any layers can follow the CNN – we just need to be able to compute their gradient.

In terms of the computational complexity of this algorithm, we can see that it mostly depends on the size of the second part of the neural network. If most of the computation in the forward pass is done in the convolutional layers, which are skipped in the explaining process similarly to CAM in section III-B, the explaining backward pass will be relatively quick. On the other hand, calculating the gradient may slow down the explaining process when complex feed-forward or recurrent networks are added to the system. However, it is crucial to remember that these models are built mostly with their predicting accuracy in mind. In reality, the model structure would rarely be changed to make the focus map generation quicker.

Guided backpropagation: A big problem of the simple Grad-CAM algorithm is the lack of detail in the final focus map – it has the same resolution as the channels of layer A, which is usually significantly lower then the input image. The authors tackled this problem by combining Grad-CAM with another algorithm originally used for visualizing the weights learned by a convolutional NN, Guided backpropagation described by Springenberg *et al.* [21]. This resulting focus map, obtained as an element-wise multiplication of Grad-CAM[4] and Guided backpropagation maps is then called Guided Grad-CAM.

Guided backpropagation is essentially just a modification of the classical backpropagation algorithm used in the backward pass when training any neural network. Assuming a nonlinear activation function $\text{ReLU}(x) = \max(0, x) = (x > 0) \cdot x$, the gradient used in classical backpropagation is either kept the same or set to 0, based on whether the original function in the forward pass contained negative numbers. In Guided backpropagation, this approach is modified by setting the gradient value to zero in other cases as well. Specifically, if the gradient value in the previous layer is negative, it is also set to zero. This can be described as taking the value from classical backpropagation and applying the ReLU function to the result. When labeling ReLU as the i-th layer of the network $\mathcal{L}^{(i)}$ and its respective gradient in backward pass $R^{(i)}$, we write:

$$\begin{aligned} \mathcal{L}^{(i)} &= \text{ReLU}(\mathcal{L}^{(i-1)}) = \max(0, \mathcal{L}^{(i-1)}) \\ R_{\text{backprop}}^{(i-1)} &= (\mathcal{L}^{(i-1)} > 0) \cdot R^{(i)} \\ R_{\text{guided}}^{(i-1)} &= (R^{(i)} > 0) \cdot (\mathcal{L}^{(i-1)} > 0) \cdot R^{(i)} = \text{ReLU}(R_{\text{backprop}}^{(i-1)}) \end{aligned} \tag{11}$$

While not costing computationally much more than classical backpropagation, significantly better results in interpretable visualization are achieved, as more of the unimportant areas are forced to zero values. However, this algorithm struggles with the same problem as other previously mentioned gradient methods in section III-A – highlighted areas in focus map do not necessarily mean a positive effect on the model prediction that we are trying to visualize. However, by multiplying this detailed but inaccurate focus map with the Grad-CAM focus mapresults in a significant improvement over both of the original models. This process is also visualized in the original illustration of Grad-CAM 3.

When discussing the computational complexity of the entire Guided Grad-CAM algorithm, it is important to realize that the gradient of the entire network needs to be computed. Therefore, the mentioned optimization for Grad-CAM with more convolutional layers and less feed-forward or recurrent layers would not be effective here. Due to combining two of the

[4]Note that the Grad-CAM focus map needs to be rescaled properly to fit the dimensions of the input image.

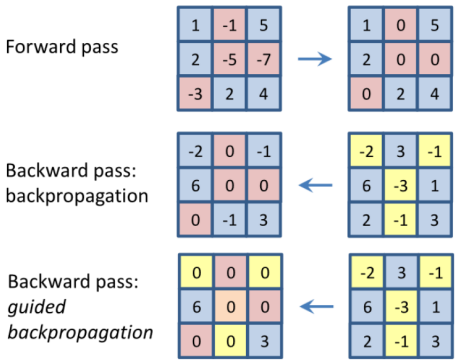

Fig. 4. Visualization of Guided backpropagation, illustrated by Springenberg *et al.* [21]. The image was modified for our purposes.

previously mentioned approaches, gradient approach and class activation mapping, this algorithm will undoubtedly be slower than any previous focus mapping algorithm mentioned. This drawback is balanced by the quality of the focus map, which is in full resolution and should reflect the actual areas affecting the model decision positively.

IV. Transparent models

Previously mentioned techniques of explainable machine learning are used for explaining already existing black box models after the decision is made (post-hoc). This approach is usually prioritizing the predicting accuracy over the explanation quality (as long as the model complexity is not reduced in any other way), as mentioned in section I-A.

As opposed to the idea of complex entangled methods such as neural networks, research was done in simpler transparent models, which strive for explainability from the beginning. Using these models, we can typically use its internal parameters used for prediction to directly explain made predictions.

A. Linear model

The linear model can be considered one of the simplest examples. While not being very advanced, it is a common basis for other transparent models, expanding it non-trivially. It is also a popular and common model in statistics, where the focus on model explainability is usually higher than the focus on accurate predictions.

For the vector of n numeric inputs $\mathbf{x} = (x_1, \ldots, x_n); \mathbf{x} \in \mathbb{R}^n$, prediction is made by calculating the dot product with the pre-trained vector of weights $\mathbf{w} = (w_1, \ldots, w_n)$ and adding the bias term w_0[5]. Single number $\hat{y}$ is then considered the output of this model as the estimate of the random variable y.

$$\hat{y} = \mathbf{x}^T\mathbf{w} + w_0 \tag{12}$$

For the true value of y, we can imagine that it consists of two factors, the known $\hat{y}$ and the unknown represented as a random variable $\varepsilon \sim \mathrm{N}(0, \sigma^2)$.

$$y = \mathbf{x}^T\mathbf{w} + w_0 + \varepsilon; \quad \varepsilon \sim \mathrm{N}(0, \sigma^2) \tag{13}$$

Weight parameters are usually learned in an explicit formula from the matrix of m training data points $\mathbf{X} \in \mathbb{R}^{m,n}$ and a vector of their reference output values $\mathbf{y} \in \mathbb{R}^m$ by solving the normal equation (14), optimizing the residual sum of squares.

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{14}$$

Thanks to this explicit formula for training the model, this process is significantly faster than an iterative learning algorithm, even though the matrix inversion is generally hard to compute. In cases when the $\mathbf{X}^T\mathbf{X}$ matrix is not invertible, meaning that some of the inputs are linearly dependent on each other, Moore–Penrose inverse with similar computation complexity is commonly used for this task. Alternative training methods introducing regularization (such as ridge regression or Lasso [22]) are also available, with the predicting part of the model staying the same.

A clear connection between the learned parameters and the output is visible from the formulas. For more significant inputs, their weight will be higher in absolute value, meaning that it can affect the output more[6]. Sign of the weight can tell us whether the correlation between the respective input and output is positive or negative.

[5]The bias term w_0 is also commonly called intercept. It is sometimes considered an element of $\mathbf{w} = (w_0, w_1, \ldots)$, then $\mathbf{x}$ is extended as $(1, x_1, \ldots)$ and the whole equation can be simplified as $\hat{y} = \mathbf{x}^T\mathbf{w}$. We will avoid this notation due to the necessity to modify input $\mathbf{x}$.

[6]However, when interpreting the model, the original meaning of the inputs must not be forgotten – the variance of the inputs also affects the weights.

B. Generalized linear model (GLM)

The idea to combine the basic linear model with other statistical models is nothing new [23]. In some cases, this model generalization can provide significantly more explaining power, while still being intuitively interpretable. The underlying idea of this model is the problem of linearity in certain use cases. We commonly have implicit requirements on the range of the predicted variable. For example, many physical quantities only make sense when positive. A purely linear model does not respect these limitations, therefore adding a non-linear factor is necessary. This effectively means we change the previous assumption of predicted variable normality – instead, we can model it using various other distributions.

Using the notation from earlier (section IV-A), we will construct a linear predictor $\eta = \mathbf{X}^T\mathbf{w} + w_0$. As opposed to equation 12, η is a linear combination of random variables $\mathbf{X} = (X_1, \ldots, X_n)$ instead of the observed values. In addition, we provide a random variable Y with mean μ, simulating the output prediction; the modeled distribution function of Y is determined by our knowledge of the domain. Link function g is used to connect the predicting variable Y with the linear predictor η:

$$g\big(E[Y|\mathbf{X}]\big) = g(\mu) = \eta = \mathbf{X}^T\mathbf{w} + w_0 \tag{15}$$

The inverse of the link function g^{-1} is also sometimes called the mean function as it helps us calculate the mean μ from the linear predictor η. In case the chosen distribution for Y needs multiple parameters, and the mean estimate is not enough for constructing the variable, an additional function V is chosen to estimate variance so that var $Y = V(\mu)$.

Training process: Due to the nature of the link function, we cannot use an explicit method in the training process, iterative methods have to be used. Iteratively reweighted least squares (IWLS) is one of the methods commonly used for this purpose. After setting an initial estimate $\mathbf{w}^{(0)}$, a sequence of estimates approaching the ideal weights can be calculated with the following formulas:

$$\begin{aligned}
\eta^{(t+1)} &= \mathbf{X}^T\mathbf{w}^{(t)} + w_0^{(t)} \\
\mu^{(t+1)} &= g^{-1}(\eta^{(t+1)}) \\
\forall i \in \{1, \ldots, n\} : W_i^{(t+1)} &= \left(\frac{\partial \mu_i}{\partial \eta_i}\left(\eta_i^{(t+1)}\right)\right)^2 \frac{1}{V(\mu_i^{(t+1)})} \\
\mathbf{z}^{(t+1)} &= \eta^{(t+1)} + (y_i - \mu^{(t+1)}) \cdot \frac{\partial \mu_i}{\partial \eta_i}\left(\eta_i^{(t+1)}\right) \\
\mathbf{w}^{(t+1)} &= (\mathbf{X}^T\mathbf{W}^{(t+1)}\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{W}^{(t+1)}\mathbf{z}^{(t+1)})
\end{aligned} \tag{16}$$

First of all we calculate linear predictor $\eta^{(t+1)}$ and mean estimate $\mu^{(t+1)}$ using the weights from the previous iteration. Then, matrix of "working weights" $\mathbf{W}^{(t+1)} \in \mathbb{R}^{n,n}$ and vector of adjusted response variables $\mathbf{z}^{(t+1)} \in \mathbb{R}^n$ are constructed. These are afterward used to compute the adjusted weights $\mathbf{w}^{(t+1)}$. This process is repeated either for a set amount of steps or until $\mathbf{w}^{(t)}$ starts converging.

After the training process is finished, we can calculate an estimate $\hat{y}$ for observed values $\mathbf{x} = (x_1, \ldots, x_n)$:

$$\hat{y} = g^{-1}(\mathbf{x}^T\mathbf{w} + w_0) \tag{17}$$

Coefficient interpretation: While the coefficient interpretation may not seem as simple as in the case of a basic linear model due to the additional processing by the link function, after second thought the conclusions turn out to be quite intuitive. However, it is important to note that the interpretation varies based on the chosen distribution modeling Y. Logistic regression is one of these special cases, modeling Y with Bernoulli distribution. Therefore, logit link function $g(p) = \ln\left(\frac{p}{1-p}\right)$ is used. In this case, the output of the entire model represents a probability and therefore needs to stay in the $[0, 1]$ interval, which is ensured by the logit link. When interpreting the model coefficients, rather than describing the effect on probability $p \in [0, 1]$, we will describe the effect on *odds* $\frac{p}{1-p} \in [0, \infty)$, which are still intuitively understood thanks to their use in sports betting.

When interpreting coefficient w_i in a trained model, we utilize the inverse of the link function g^{-1}, which in the case of logit is the logistic function $\frac{1}{1+e^{-x}}$. We can notice that by changing any of the inputs x_i by 1, the predicted odds are multiplied by the factor of e^{w_i}, therefore odds increase if and only if $w_i > 0$. Compare this interpretation with the simple linear model, where a constant of w_i would be added to the output. An analogous process can be done for any other link function, with a clear connection between the model weights and output change mapping as its result.

C. Generalized linear rule model (GLRM)

Generalized linear rule model [24], abbreviated as GLRM, is another interpretable technique to create transparent models. To have an overview of GLRMs, they extend the GLMs mentioned in the previous section IV-B by introducing rule-based modeling to the algorithm. While GLMs use the linear combination of raw features, GLRMs use a set of rules derived from the data. The perk of these models is being naturally interpretable and, simultaneously, able to represent complex, nonlinear associations with the rules formulated [3]. In the paper where the model was originally described [24], Wei *et al.* refer to

rules as rule ensembles based on the ensemble methods in machine learning since they are the collection of individual rules assembled to construct a predictive framework. They carried out experiments applying linear and logistic regression, two models of GLMs. Before examining the results of those experiments, the computational aspects of GLRMs should be mentioned.

Wei *et al.* [24] used GLMs when formulating GLRMs. The main difference is defining a rule space and getting their linear combination. In the supervised setting, there is a target variable $y \in \mathcal{Y}$ and features $\mathbf{X} = (X_1, \ldots, X_d)$ creating a dataset of n instances that are independent and identically distributed, each of which represented by $(\mathbf{x}_i, y_i)$ where $i = 1, \ldots, n$. They allowed $\mathcal{K}$ to represent the group of conjunctions associated with $\mathbf{X}$ to be examined. It relates to a specific approach for formulating the rules underpinning the model. It is important to emphasize one crucial presumption they made: each feature X_j within $\mathbf{X}$ has been converted to binary form. This paves the way for applying the conjunction operations on the features and generating the rules. In the final step, the variable A_k should also be introduced before presenting the formulations, denoting the conjunction $k \in \mathcal{K}$. For every instance in the dataset, A_k takes binary values ($a_{ik} \in \{0, 1\}$ for sample i), indicating if a data point adheres to the conditions set by the corresponding rule.

$$p_{Y|\mathbf{X}}(y|\mathbf{x}) = h(y) \exp(\eta y - \Phi(\eta)) \tag{18}$$

$$\eta = \sum_{k \in \mathcal{K}} \beta_k A_k \tag{19}$$

$$\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} \left[\Phi \left(\sum_{k \in \mathcal{K}} \beta_k a_{ik} \right) - y_i \sum_{k \in \mathcal{K}} \beta_k a_{ik} + \sum_{k \in \mathcal{K}} \lambda_k |\beta_k| \right] \tag{20}$$

The above formulations presented build the fundamentals of GLRMs. Equation 18 stands for the probability distribution function pertinent to GLMs. GLRM's difference comes into play in 19, where the canonical parameter η has been defined using the conjunctions A_k and learned parameters β_k. The optimization problem presented in equation 20 seeks to minimize the negative log-likelihood function to conclude the coefficients β_k in 19. A final important note here is that the $\phi(\eta)$ function can be replaced with the function $\log(1 + e^{\eta})$ in the case of logistic regression and with $\eta^2/2$ in the case of linear regression.

It is mentioned in the paper that the fundamental difficulty in building rule-based models stems from the exponentially large number of conjunctions generating the rule space. [24] That makes exploring all possible rules for datasets with many features practically infeasible. To tackle this problem, they proposed two approaches.

First-degree rules: In the first-degree rule approach, the idea is to simplify the rules. It is achieved by involving only a single feature per rule. So, multiple feature interactions are absent from the composition of the rules, meaning the conjunctions A_k are equivalent to the features X_j that are in the binary form. This leads to a significant reduction in computational complexity and a more straightforward computation. Furthermore, regarding interpretability, it can be said that using the first-degree approach offers greater interpretability since it represents the direct relationships in the data. Nonetheless, being more interpretable can make the model unable to capture complex relationships, a typical issue in the trade-off between interpretability and performance.

Column generation: Their idea of using the column generation technique is based on solving the reduced version of the problem instead of using the whole rule set with all of the possibilities. The solution starts with using a subset of rules in the initial step in the context of GLRMs. The next step is determining the rules out of the initial subset that can potentially improve the objective function. Then, these selected rules can be added as new rules to the initial subset. This iterative process stops when adding more rules doesn't further the objective function. In terms of complexity, not attempting to process all possible rules simultaneously provides efficiency and feasibility. The extent of its effect on interpretability correlates with the number of rules chosen during these iterations.

In the practical setting, Wei *et al.* [24] conducted two experiments where various machine learning methods were tested, one for classification and one for regression. For the classification task, they compared logistic regression with first-degree rules and logistic regression with column generation. For the regression task, the same setting has been considered with linear regression. Their experiments also include gradient-boosted classification and regression trees and support vector machines. Based on the results, they observed that the models utilizing the column generation functionality often result in more favorable trade-offs when the dataset is extensive. However, in a small dataset, there is a possibility that the models with column generation face an overfitting issue. In addition, when the maximum performance is aimed, the models with column generation, particularly the ones that include the numerical features of the dataset, exhibit strong competitiveness when measured against benchmark models. Furthermore, in the context of the methods put forward, it's noteworthy that using numerical features in the data is more advantageous in regression applications than classification.

D. Generalized additive model with pairwise interactions (GA2M)

The last model of this section builds upon the foundation of generalized additive models (GAMs) [25]. Generalized additive model with pairwise interactions, abbreviated as GA^{2}M, proposed by Lou *et al.* [26], are tailored to capture not only the individual effects of features in the data but also their pairwise interactions in a predictive setting. From an interpretability

standpoint, since each feature's impact on the prediction can be distinctly determined, GA^{2}Ms are considered inherently interpretable, deriving meaningful understanding from the forecasts [3]. For a better understanding of GA^{2}Ms, GAMs should be explained briefly.

$$g(E[y]) = \sum_i f_i(x_i) \tag{21}$$

$$g(E[y]) = \sum_i f_i(x_i) + \sum_{i,j} f_{ij}(x_i, x_j) \tag{22}$$

$$\min_{F \in \mathcal{H}} E[L(y, F(\mathbf{x}))] \tag{23}$$

In their paper, Lou *et al.* [26] first provide equation 21 for GAMs, g being the link function, y being the outcome, x_i being the i-th feature and f_i being the i-th smooth function. Based on the formula, it can be deduced that their interpretation is straightforward since they only consist of terms involving only one variable. However, this attribute of GAMs restricts capturing complex interactions in the dataset. So, they proposed GA^{2}Ms as an extension to capture more complex interactions pairwise while maintaining the interpretability aspect. Equation 22 presents the basic formula for GA^{2}Ms. From the computational perspective, the main challenge they refer to is having an extensive array of feature combinations to examine. To briefly define the problem, they used the notation $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ to describe the problem's dataset. In this notation, $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,n})$ stands for the input features, y_i for the target variable, n for the number of features, and N for the number of samples in the dataset. In addition, to represent the indices for all features in the dataset, they used $\mathcal{U} = \mathcal{U}^1 \cup \mathcal{U}^2$, where $\mathcal{U}^1 = \{\{i\} | 1 \leq i \leq n\}$ and $\mathcal{U}^2 = \{\{i, j\} | 1 \leq i < j \leq n\}$. By employing the Hilbert space $\mathcal{H} = \sum_{u \in \mathcal{U}} \mathcal{H}_u$ and its additive group of functions $F(\mathbf{x}) = \sum_{u \in \mathcal{U}} f_u(x_u)$, they derived formula 23, representing the objective function that needs to be minimized based on the given notation. In formula 23, L stands for the loss function. Their iterative algorithm for GA^{2}M begins with two interaction sets and the best model from the Hilbert space. While one of the sets contains the chosen pairs so far, the other contains the remaining ones. Next, the algorithm creates a model for each feature pair that is not chosen, selects the best pair, and adds it to the set containing chosen pairs. This procedure is continued until an improvement in accuracy is no longer observed. From the computational perspective, the main challenge they refer to is having an extensive array of feature combinations to examine. Addressing this issue, they formulated a novel approach called FAST, focusing on finding and systematically ranking all interactions among feature pairs. This approach enabled them to reduce the number of pairwise interactions used in prediction from $O(n^2)$ to $O(n)$, significantly reducing the computational complexity of the prediction process.

An important work on GA^{2}Ms has been done by Caruana *et al.* [27]. They conducted two case studies, pneumonia risk prediction and hospital 30-day readmission. The results show that GA^{2}Ms can reach an accuracy level comparable to state-of-the-art methods while maintaining their intelligibility, which is vital for applications of critical importance.

V. Related work

Multiple different methods of XAI were brought up, often related to previously mentioned topics. We provide a basic summary of the methods we encountered when writing this article.

A. SHapley Additive exPlanations (SHAP)

In addition to the methods presented in section II, Shapley Additive Explanations or SHAP, proposed by Lundberg *et al.* [28], should also be mentioned. In the related paper, they performed a unification of several methods containing LIME [4] and DeepLIFT [29], [30]. Their inspiration comes from the cooperative game theory: The method provides a way to apportion the prediction among the features. This has been executed through additive attribution methods, allocating a significance value to each feature. One of the critical properties of the technique is its consistency. They state that should a change in the model result in a feature's heightened or reduced role, the importance designated to that feature should correctly reflect this variation.

In the taxonomy created by Linardatos *et al.* [3], SHAP falls under the category of model-agnostic methods since it is applicable across various model types. In terms of scope, it focuses on local explanations, but their paper shows that it is also possible to obtain global explanations.

B. Anchors

Anchors, proposed by Ribeiro *et al.* [31], is another model-agnostic method focusing on local explanations. The model utilizes highly precise rules based on conditions that keep the model's prediction stable. The algorithm is tailored to effectively produce explanations, determining the essential features contributing to the model's prediction for a given instance. It can be said that high-precision rules suggest that the model's output is uniform for similar cases. The model's flexibility was tested across diverse datasets with various models, illustrating that Anchors can successfully reveal the input segment necessary for the classifier's decision-making. As a final note, they also mentioned some limitations. They stated that the rules generated for

the predictions that lie close to the decision boundary of the model perform poorly in generalizing new instances. Furthermore, there is a chance that there will be a conflict between generated anchors. In the end, the last limitation they presented is that there are situations where the generated anchors might give rise to complicated explanations.

C. PatternNet, PatternAttribution

Another set of methods to explain images with focus maps was provided by Kindermans *et al.* [32]. The essential idea was to separate the input data $\mathbf{x}$ to two components, signal $\mathbf{s}$ and distractor/noise $\mathbf{d}$, such that $\mathbf{x} = \mathbf{s} + \mathbf{d}$. Only the signal part is useful for making a prediction, whereas the distractor complicates our prediction. In simple cases of linear models, the distractor is commonly considered a vector of normally distributed variables (see section IV-A). However, in practical cases, we do not know the values of $\mathbf{s}, \mathbf{d}$ and we can only get approximations. Authors provide innovative ways to estimate the signal, which is then projected back into input space in the PatternNet method and serves as a focus map.

In PatternAttribution, the elementary idea is to measure so-called attribution, showing how much the output is affected by different signal dimensions, meaning pixels in the case of the image. The relevances across all input color channels must be merged into one by averaging to visualize the output as a focus map. The attribution is computed as an element-wise vector multiplication of weights $\mathbf{w}$ and a vector of parameters $\mathbf{a}_+$ from the two-component signal estimator. As a result, an even more precise focus map is generated in most cases, which comes at the cost of losing information about the input color channels and higher computational complexity.

D. Human attention

The human attention approach was properly utilized by Das *et al.* [14] when working with neural networks performing visual question answering. Human attention dataset (VQA-HAT) has been created, matching the pairs of an input image and a regarding question to a focus map generated from human attention. This dataset can be used for multiple purposes: either as a training source for a new prediction model, potentially increasing model prediction accuracy, or to compare these maps with explanations made by an existing model. If a deep learning model makes predictions based on the same image regions as human subjects, we intuitively tend to trust its results more.

The authors used multiple techniques when generating these focus maps. The Human Control Interface was intentionally chosen to be simple; tested subjects uncovered parts of a blurry image using a computer cursor. Authors have shown that human behavior in this test differed when the subjects were shown the VQA task and the textual answer. In tests where the answer was supposed to be provided by tested subjects, sometimes subjects uncovered too little of the image and provided an incorrect answer. In another method, subjects were shown the entire uncovered image with the question and were supposed to select important parts.

VI. Conclusion

Many ongoing works in explainable AI focus on the different aspects of explainability at different parts of the deep learning workflow. In this paper, we have presented the different approaches to explainability and methods to achieve it. Explainability is a critical factor in adopting deep learning models in real-world applications. Direct explainability methods, such as the mentioned transparent modifications of the linear model, produce more complete explanations but are limited in their scope. On the other hand, methods like Grad-CAM are more flexible; this comes at the cost of challenging connection to the inner workings of these models. Unlike completeness, the interpretability aspect of an explanation depends more on the described problem, rather than the used explaining model. An interpretable explanation needs to be simple enough to be comprehensible by a human, all while providing just enough information to explain anything. For complex problems with many input features, this task may be complicated or even unrealistic. Another important aspect of interpretability is the final presentation/visualization of an explanation, which can highly affect how comprehensible it is for humans.

The choice of explainability method should depend on the application and the level of impact of the model's decision on the possible outcomes and the dangers of a wrong decision. Study of explainability has shown that the representations learned by the model might occasionally share semantic information with human values and understanding. Models that learn explainable representations are more fit for use in highly critical systems that affect human life, such as law or medicine.

Additionally, we have highlighted the computational aspects of these methods. Every model differs in their requirements for computational resources, which influence their applicability in real-world scenarios regarding feasibility and efficiency. The computational burden also plays a role in the scalability of these methods. Investigating these computational factors allows a deeper understanding of the trade-off between explainability and performance.

In order to summarize the described models concisely, we have created a summarizing table I that contains relevant information about these explainers.

TABLE I
DESCRIBED MODEL SUMMARY

Model name	Predicting model	Common task	Local/global	Accuracy/explainability tradeoff
LIME	Any	Any	Local	Balanced
Disentangled representations	Any	Any	Local, global	Balanced (with the right level of disentanglement)
Gradient methods	Neural network	Image classification	Local	Prediction accuracy
Class Activation Map	Convolutional NN	Image classification	Local	Balanced
Grad-CAM	Neural network	Image classification	Local	Prediction accuracy
Linear model	Linear model	Data vectors	Global	Explainability
Generalized LM	Generalized LM	Data vectors	Global	Explainability
GLRM	GLRM	Data vectors	Global	Balanced
GA^2M	GA^2M	Data vectors	Global	Balanced

ACKNOWLEDGMENT

This article was written as a part of the Computational Aspects of Machine Learning Seminar[7] at the Technical University of Munich, School of Computation, Information and Technology, Chair of Scientific Computing. We want to thank Iryna Burak from TUM (iryna.burak@tum.de) for supervising us in the writing process and providing valuable insight. We would also like to thank our fellow students at TUM for writing their peer reviews anonymously and Eliška Orsáková for proofreading the introduction section.

REFERENCES

[1] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," 2019.
[2] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (xai)," *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018.
[3] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable ai: A review of machine learning interpretability methods," *Entropy*, vol. 23, no. 1, 2021. [Online]. Available: https://www.mdpi.com/1099-4300/23/1/18
[4] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier," *CoRR*, vol. abs/1602.04938, 2016. [Online]. Available: http://arxiv.org/abs/1602.04938
[5] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2022.
[6] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," *CoRR*, vol. abs/1606.03657, 2016. [Online]. Available: http://arxiv.org/abs/1606.03657
[7] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-VAE: Learning basic visual concepts with a constrained variational framework," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=Sy2fzU9gl
[8] Y. Bengio, A. C. Courville, and P. Vincent, "Unsupervised feature learning and deep learning: A review and new perspectives," *CoRR*, vol. abs/1206.5538, 2012. [Online]. Available: http://arxiv.org/abs/1206.5538
[9] I. Higgins, D. Amos, D. Pfau, S. Racanière, L. Matthey, D. J. Rezende, and A. Lerchner, "Towards a definition of disentangled representations," *CoRR*, vol. abs/1812.02230, 2018. [Online]. Available: http://arxiv.org/abs/1812.02230
[10] C. Eastwood and C. K. I. Williams, "A framework for the quantitative evaluation of disentangled representations," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:19571619
[11] I. T. Jolliffe, *Principal Component Analysis and Factor Analysis*. New York, NY: Springer New York, 1986, pp. 115–128. [Online]. Available: https://doi.org/10.1007/978-1-4757-1904-8_7
[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385
[13] A. Agrawal, J. Lu, S. Antol, M. Mitchell, C. L. Zitnick, D. Batra, and D. Parikh, "Vqa: Visual question answering," 2016.
[14] A. Das, H. Agrawal, C. L. Zitnick, D. Parikh, and D. Batra, "Human attention in visual question answering: Do humans and deep networks look at the same regions?" 2016.
[15] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, 1998.
[16] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," 2014.
[17] Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 1, 2001, pp. 105–112 vol.1.
[18] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
[19] V. Camus, L. Mei, C. Enz, and M. Verhelst, "Review and benchmarking of precision-scalable multiply-accumulate unit architectures for embedded neural-network processing," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 4, pp. 697–711, 2019.
[20] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," *International Journal of Computer Vision*, vol. 128, no. 2, pp. 336–359, oct 2019. [Online]. Available: https://doi.org/10.1007%2Fs11263-019-01228-7
[21] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," 2015.
[22] W. N. van Wieringen, "Lecture notes on ridge regression," 2023.
[23] J. A. Nelder and R. W. M. Wedderburn, "Generalized linear models," *Journal of the Royal Statistical Society. Series A (General)*, vol. 135, no. 3, pp. 370–384, 1972. [Online]. Available: http://www.jstor.org/stable/2344614
[24] D. Wei, S. Dash, T. Gao, and O. Gunluk, "Generalized linear rule models," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 6687–6696. [Online]. Available: https://proceedings.mlr.press/v97/wei19a.html

[7]Course URL (TUMonline)

[25] T. Hastie and R. Tibshirani, "Generalized Additive Models," *Statistical Science*, vol. 1, no. 3, pp. 297 – 310, 1986. [Online]. Available: https://doi.org/10.1214/ss/1177013604

[26] Y. Lou, R. Caruana, J. Gehrke, and G. Hooker, "Accurate intelligible models with pairwise interactions," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 623–631. [Online]. Available: https://doi.org/10.1145/2487575.2487579

[27] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad, "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 1721–1730. [Online]. Available: https://doi.org/10.1145/2783258.2788613

[28] S. M. Lundberg and S. Lee, "A unified approach to interpreting model predictions," *CoRR*, vol. abs/1705.07874, 2017. [Online]. Available: http://arxiv.org/abs/1705.07874

[29] A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje, "Not just a black box: Learning important features through propagating activation differences," *CoRR*, vol. abs/1605.01713, 2016. [Online]. Available: http://arxiv.org/abs/1605.01713

[30] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," *CoRR*, vol. abs/1704.02685, 2017. [Online]. Available: http://arxiv.org/abs/1704.02685

[31] M. T. Ribeiro, S. Singh, and C. Guestrin, "Anchors: High-precision model-agnostic explanations," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/11491

[32] P.-J. Kindermans, K. T. Schütt, M. Alber, K.-R. Müller, D. Erhan, B. Kim, and S. Dähne, "Learning how to explain neural networks: Patternnet and patternattribution," 2017.