On Explainable Artificial Intelligence and Its Purposes

Adrian Duric

Spring 2024

Abstract

Alongside the current rapid development in Machine Learning (ML), the field of Explainable Artificial Intelligence (XAI) too has become a topic of increased interest. As ML models are given more responsibility in critical decision-making tasks, it becomes necessary to know why the model made the choice it did, to ensure that its reasoning is sound. Finding such an explanation is the main purpose of XAI. This essay examines what makes for a good explanation and explains how XAI techniques are used to generate such explanations for a model's decision. It also discusses how XAI may be used for purposes other than solely generating explanations, such as improving the performance of a given ML model.

1 Why Explainable AI?

In recent years, research into AI and particularly ML has exploded, as the development in the field has given vastly better-performing algorithms than what had been seen prior. Models are capable of learning from extremely large and complex data, such as images and video, audio and text, as well as the more traditionally used tabular data. With this development, new use cases for ML models have emerged, allowing the models to be used in new contexts. Gohel et al. [2021] mention wide-spanning fields of application such as healthcare and the autonomous industry, e.g. self-driving cars.

A common factor for many of these new applications is that they typically consider large amounts of data to make complex decisions. A human would not be able to consider all of the data simultaneously, the way an ML model typically does. Consider, for instance, a person driving a car in the middle of heavy traffic, and observing the road ahead. There might often be too many cars on the road and people on the sidewalk for the driver to have full awareness of each one of them. Instead, an experienced driver would likely be able to recognize which ones are important to pay attention to in a given traffic situation and drive accordingly.

If, instead, a self-driving car were to perform the same analysis of the traffic situation, an ML model would only consider a series of images, each of them consisting of an abundance of pixels. Due to how powerful the state-of-the-art (SOTA) models are, a model could be able to identify, and consider, even more objects (i.e. cars, pedestrians, etc.) than the human. Thus, it would also be able to factor each of these identified objects into its decision-making regarding where and how to drive.

However, both humans and autonomous vehicles are prone to crashing, and when a crash does occur, it becomes important to learn why; the car developers would need to learn what caused the crash to find out how they could prevent it from happening again. The human would likely be able to give some explanation as to how the crash occurred, e.g. due to loss of concentration, not noticing another car, etc. The autonomous car, however, provides little or no insight on its own. If, for instance, the crash happened because the car's ML model wasn't trained in similar environments to the one it encountered during the crash, it would be practically impossible to extract that knowledge only by examining the model. This is because black-box models (such as deep ML models) tend to be too complex for a human to understand [Gohel et al., 2021]. It is infeasible for a human to discern any information from a trained model's parameters if the model is too large, and as such, the model has no intrinsic way of explaining its actions.

This is what XAI techniques are developed for doing; they take predictions from black-box prediction models and output some explanation for how that prediction was made. In some cases, methods that are inherently more interpretable may also be used to replace black-box models entirely. Since the focus of XAI as a field lies on how to interpret models and explain their decisions, it becomes important to understand the concepts of explainability and interpretability, and what makes for a good explanation.

2 Understanding Models and Their Predictions

While there is purpose to separating between explainability and interpretability, the differences between them are nuanced, and as such, they are often used interchangeably by researchers [Linardatos et al., 2021]. The following are definitions of the terms, emphasizing these differences.

2.1 Interpretability

As Linardatos et al. [2021] put it, interpretability has to do with a human's ability to understand the causality between a prediction model's inputs, and its output, i.e. the prediction. A system with a high degree of interpretability is one where, given some input and the system output, a human would be able to point to which features of the input were the most pivotal in causing the system to output the prediction it did. Imagine having some arbitrary model with high interpretability; what follows from this definition is that understanding the model itself may in some cases be irrelevant–one may be able to interpret the model's prediction from looking at the inputs and outputs alone, ignoring the black-box layer in between. This is exemplified in 3.3, as model-agnostic methods work precisely by only regarding the input and output, decoupling them from the underlying model [Molnar, 2022].

2.2 Explainability

When compared to interpretability, the concept of explainability differs in that explainability also concerns understanding the inner workings of a given prediction model. A highly explainable model is one where a human can comprehend its composition and understand the logic behind how it makes all its predictions.

When considering deep neural networks, researchers may understand them on an overarching, conceptual level, but they are generally considered difficult to explain. Consider a Convolutional Neural Network (CNN). It is possible to explain what a single convolutional layer does thoroughly, but as the CNN grows in depth, it becomes impossible for humans to follow exactly what the model is learning to recognize from hidden layer to hidden layer, which of course also varies depending on the training data used. The model may still be interpretable using XAI methods, but not necessarily easily explainable.

On the other hand, other models exist that are intrinsically easier to understand, and thus also easier to explain, to humans. Some examples are intuitively comprehensible methods such as linear regression and K-nearest neighbors, as discussed in 3.2.

2.3 What Makes an Explanation Good?

When evaluating the performance of ML models, there are many commonly used measurements, such as accuracy, precision, recall, and other variants. The commonality between these measures is that they are based on calculating statistics for how often observations of specific types were classified correctly or falsely, or classified to some specific label, or similar. In general, most of these measurements tend to be quantitative.

Evaluating an explanation, on the other hand, is more qualitative. Molnar [2022] lists several important definitions to help evaluate the quality of an explanation. The following are some of the most important ones:

- **Fidelity** concerns how well an explanation fits to a given prediction from a black-box model. It has close ties with the accuracy of the model itself because, as Molnar [2022] puts it, ”if the black box model has high accuracy and the explanation has high fidelity, the explanation also has a high accuracy”, meaning that the explanation alone would also work well to predict an outcome on the same, unseen data that the model gave a prediction for. An explanation with low fidelity would be one which fails to explain the cause of a prediction.
- **Consistency** refers to how consistent the explanations would be from two models trained on the same data and giving similar predictions. If the two models emphasize the same features in the input data, they should also have similar explanations, meaning high consistency would be desirable. However, they may also reach the same prediction by emphasizing different features. In this case, the explanations for the two predictions should differ, meaning high consistency is not always desirable for explanations; it depends on the causality behind the predictions.
- **Stability** can perhaps be considered the counterpart to consistency; stability refers to a single model’s ability to provide similar explanations for similar inputs. This is a critical component not only to assess the quality of explanations but also model predictions in general, meaning the concept of stability is also used to evaluate ML models, not only XAI methods. A notable example of how stability has been tested is that of adversarial attacks, as demonstrated by Goodfellow et al. [2015].
- The **degree of importance** of features in the input may or may not be transparent from the output of a prediction model. A good explanation should comment on which features were significant in leading to some prediction, and which ones were less relevant. Some simpler models like linear regression models give these explanations naturally, as discussed in 3.2. For other more complex models, like deep neural networks, it is typically impossible for a human to acquire information about feature importance from the model alone. For these cases, different methods must be used, such as saliency maps for CNNs, as well as others mentioned in 3.3.
- Finally, the arguably most important quality of an explanation is its **interpretability**. This, of course, presumes that the intention of explaining something is for the receiver of the explanation to understand it better. What makes for an interpretable explanation is hard to define, as different people may perceive the same explanation in different ways. Therefore, it is important to give explanations that fit the audience. A layman may struggle to understand a long and complex explanation, and an expert may be frustrated at the lack of details in a simpler one. If intended for a machine, an interpretable explanation could mean something else entirely.

3 Methods in XAI for Generating Explanations

Now having defined several metrics of evaluation for explanations, this section presents a range of methods used in the field of XAI to produce explanations for predictions. It is not an exhaustive list but is rather meant to introduce some of the most widely used ones, as many other methods are derivative of these.

3.1 Taxonomy of XAI Methods

Many taxonomies have been proposed for how to discuss inherent differences between methods. Sometimes it is useful to have context-specific terminology; for instance, Wahde and Virgolin [2023] utilize the 5 principles of *interpretability*, *inherent capability to explain*, *independent data*, *interactive learning*, and *inquisitiveness* in the context of evaluating transparency and accountability for conversational AI. Yet still, many general XAI surveys make similar distinctions [Das and Rad, 2020] [Gohel et al., 2021] [Linardatos et al., 2021]. Das and Rad [2020] sort methods by considering their *scope*, *methodology*, and *usage*.

- **Scope** refers to whether the method explains specific predictions (or groups of predictions) made by a model, or rather explains the inner workings of the model itself. The former methods have a *local* scope, whereas the latter have a *global* scope.
- **Methodology** refers to the algorithm itself, i.e. how the method works. Specifically, the distinction made here is between methods focusing on the input data to the prediction model, and methods that use the prediction model's parameters to produce their explanations. In the first case, this is referred to as *perturbation*. Several models work by tweaking the inputs in some random or deliberate way and looking at how that change affected the prediction of the model in question. The other case is related to *backpropagation*, as these methods typically consider gradients of neural networks during backpropagation to derive information about how the input affected the parameters.
- **Usage** refers to how the method is being used; *intrinsic* methods are those that are dependent on the architecture of the prediction model. In some cases, such methods are integrated into the model, given that the method is compatible with a given model. Other methods are essentially models of their own, that by design are human-interpretable without any further modifications. The latter distinction is what Molnar [2022] refers to as *interpretable models*, though these are still a subgroup of the intrinsic methods. Besides intrinsic methods, there are also *post-hoc* methods, also referred to as *model-agnostic methods*. These are methods that are independent of the model architecture and thus can be applied to any prediction model.

These and other labels may have some overlap, in the sense that certain models may fit multiple labels at once, sometimes conflicting ones. For instance, a simple linear regression model can be said to have a local scope, as explains a given sample. At the same time, it also has a global scope, as the model itself is easily interpretable by reading off the parameters belonging to each feature. The point of these labels is thus not to perfectly sort all methods into groups of labels, but rather to recognize similarities between methods with the same labels.

To go through the methods methodically, I have chosen to divide the methods into categories similar to those proposed by Gohel et al. [2021]. However, I have chosen to label some methods differently; for instance, in both [Gohel et al., 2021] and [Das and Rad, 2020], saliency maps and related methods are labeled as model-agnostic methods. However, as Molnar [2022] points out, such algorithms are dependent on parameters and gradients found in neural networks, and as such are not truly model-agnostic, though they can be used on a wide variety of ML models. Nevertheless, I believe the mechanisms behind each method will make their usage clear.

3.2 Interpretable Models

As mentioned, interpretable models are models that are designed in such a way that humans can intuitively understand them. While also prevalent in the domain of AI, several of these methods are also commonly found in other fields, such as more general statistics.

3.2.1 Regression Models

Linear regression is a classical technique in statistics and concerns how to fit a linear function to a dataset to best represent the data. This function is expressed as

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_d x_d + \epsilon \tag{1}$$

where y is the predicted value, $x_i, i = 1, \ldots, d$ is the value of the ith feature of the input vector x (x is d-dimensional), β_i is the regression coefficient of the ith feature, and ϵ is the error term.

Once the linear regression model is fitted to the data, the matter of explaining predictions through it is very straightforward; one simply looks at the regression coefficients. They explain exactly how much each feature of the sample contributes when producing the output. As explained previously, this gives an explanation on both a local and a global level; when the model is fitted to the data, the linear function expresses the line that represents the data in the most accurate way possible. If the dataset represents the population it was drawn from well, then the fitted regression coefficients will also accurately represent the effect a given feature tends to have on samples from the population.

Other related regression models are also commonly used. For classification tasks, **logistic regression** can be used instead of linear regression. The way of understanding explanations from the logistic regression model is analogous to the linear case. There are also more flexible, yet also more complex models in existence, such as **Generalized Linear Models (GLMs** and **Generalized Additive Models (GAMs)**. GLMs can, for instance, improve on some weaknesses of linear regression such as dealing with confounding variables, and GAMs even allow for non-linearity.

3.2.2 Decision Trees

Decision trees are another popularly used technique for classification. There are several ways to construct a decision tree, but importantly, they follow the same basic structure when completed. Given a dataset of d features per instance x, the decision tree asks sequential questions about specific feature values of x to determine a class to classify x to. The questions function as nodes in the tree. An answer corresponds to an edge or a branch in the tree, and each branch leads down a separate path until reaching a leaf node, corresponding to the class x belongs to. The feature values asked about when classifying x are determined in the creation of the tree, and are dependent on feature values present in the dataset used to create the tree.

Like regression models, decision trees can both be interpreted on a global scale (i.e. the entire model is interpretable) and also deliver explanations to individual explanations on a local scale. The global interpretation comes from looking at the nodes, or questions, as they tell which features are evaluated to determine which class, or group of classes, the input can belong to. The tree also naturally gives an idea of feature importance, as the features used in the nodes furthest up in the tree are especially significant. On the local scale, the explanation of a prediction comes from looking at all the questions asked when classifying a given sample. The questions tell which feature values were considered in performing the classifications.

3.2.3 K-Nearest Neighbors

Conceptually, k-nearest neighbors (kNN) is not a parametric, trainable model like regression models or decision trees. It is an algorithm that, given a dataset, classifies a sample x simply by checking the classes of the k nearest neighbors and selecting the majority class. k is a hyperparameter that can be chosen as desired, to adjust how sensitive the prediction is to local trends in the vicinity of x. Since there are no model parameters, the entire interpretation of kNN comes from understanding the dataset. Essentially, this means knowing where there are local trends, or clusters, among data points in the dataset, and seeing where a new sample is placed compared to these clusters.

This method could also have been featured in 3.3, as it is precise to call it model-agnostic. However, the typical way that model-agnostic methods are used is to couple them with uninterpretable models, to make them easier to interpret. kNN works independently of other models, and in terms of explainability, it is rather easy for humans to understand kNN algorithmically, a common trademark of interpretable models and methods.

3.3 Model-Agnostic Methods

In comparison to 3.2, we typically talk about model-agnostic methods not as models of their own meant to replace a less interpretable model (as can be the case for interpretable models), but rather as algorithms that complement existing models to make them more interpretable. As a consequence, model-agnostic methods tend to rely on perturbation of the input, as opposed to backpropagation in terms of methodology. As discussed in 3.1, relying on a model to function like a neural network with gradients to update parameters makes an XAI method model-specific.

3.3.1 Surrogate Models

Having already proclaimed the general difference between interpretable models and model-agnostic methods, the main idea of surrogate models as a model-agnostic method is, ironically, to replace an uninterpretable model with an interpretable one. However, there is a major difference here; parametric models are typically fed a dataset of labeled data, and based on the data, they learn to predict among the same labels on unseen data. The idea of surrogate models is that the data used to train the interpretable model is labeled with whichever label the complex model predicts, instead of the actual true label. That way, the interpretable model does not learn to represent the dataset itself; it learns to represent the complex model(!).

This method brings all the advantages of whichever surrogate model is chosen. Examples of these are the models mentioned in 3.2, such as linear regression or decision trees. Of course, the fact that the surrogate learns to imitate another model gives a disadvantage; the surrogate model will likely not fit the other model's predictions precisely. Rather, if the other model is very complex, the surrogate may be far off with its predictions, due to inductive bias. Furthermore, if the complex model itself performs poorly, this would likely make the surrogate's performance even worse, giving its interpretability little worth if the predictions it makes are wrong.

3.3.2 Local Interpretable Model-Agnostic Explanations

Considering 3.3.1 from a taxonomic perspective, surrogate models can be said to have a global scope; they attempt to replicate another model while still being intrinsically interpretable, thus explaining the model they replace. Local interpretable model-agnostic explanations (LIME), proposed by Ribeiro et al.

[2016], is based on surrogate models too, but has a local scope. Mathematically, the algorithm is described through the following equation:

$$\xi(x) = \underset{g \in G}{\arg\min}\ L(f, g, \pi_x) + \Omega(g) \tag{2}$$

where:

- x is the sample to be explained,
- G is the set of interpretable models (so $g(x)$ could be linear regression, decision trees, etc.),
- $f(x)$ is the model being explained by g,
- $\pi_x(z)$ is a proximity measure between x and another instance z,
- L is a loss function measuring how far off g is from approximating $f(x)$ in the vicinity of x defined by π_x, and
- $\Omega(g)$ is a function giving a numerical measure of the complexity of g, with complexity here meaning an inverse measure of interpretability. Ω may be calculated differently depending on the nature of g (e.g., for linear regression, an appropriate measure for $\Omega(g)$ may be the number of features used in the model).

The goal of the algorithm is thus to minimize $\xi(x)$ by selecting a $g(x)$ that approximates $f(x)$ well within the neighborhood of x, π_x. At the same time, the equation rewards non-complex (and therefore interpretable) functions through the $\Omega(g)$ function. Intuitively, this works by weighing samples in the vicinity of x according to π_x so that especially the closest neighbors are influential in forming a new decision boundary through g. This, Ribeiro et al. [2016] argue, gives the surrogate function g a high local fidelity, meaning it accurately represents f in the local vicinity of x. At the same time, due to the low complexity of g, the explanation provided through g should also have high interpretability simultaneously. As in 3.3.1, the explanation will derive its traits from whichever surrogate model g is chosen.

3.4 Model-Specific Methods

When thought of in comparison to model-agnostic methods, it might seem like an unnecessary drawback to use a model-specific XAI method, as it of course is dependent on the type of model it is used to explain and thus becomes potentially unusable for other models. However, the fact that model-specific methods are designed with specific model architectures in mind can also be a benefit; it means that methods are allowed to utilize model-specific information which may add to the value of the explanation produced. A very prevalent use case of this is for neural networks; while there are far too many parameters in a deep network for a human to make sense of them, they can still give valuable information to methods designed to dissect deep neural networks.

3.4.1 Occlusion

Occlusion is a perturbation-based technique that consists of occluding certain parts of the model input to learn something about how that part of the input affected the output. It is a very useful technique for uncovering the importance of specific features, or groups of them (if many are occluded at a time), and is often seen used alongside or as part of other methods. Technically, the principle of occlusion is model-agnostic, as it disregards the insides of the model it is used on and can thus be used for any model. The reason I have chosen to list it among the model-specific methods is by convention; occlusion is a commonly used technique in image classification cases and is therefore typically used on models made for image classifications such as CNNs or vision transformers.

This application of occlusion is model-specific due to how it works on image data. Zeiler and Fergus [2014] demonstrate the use of the technique by sliding an occlusion filter over an input image, setting areas of pixels to zero (dark pixels) while all the rest of the image remains as before. Both the original image and all its variations with the occlusion filter placed at different spots are passed through the already fully trained model, and the model's predictions are recorded. By looking at the differences in class probabilities for the original and occluded images, we receive information about which parts of the image add to and subtract from which class probabilities. Thus, we find which pixels and regions are influential for particular predictions.

While this method may not provide much global information about the inner workings of the model (we still do not know *why* it is influenced by certain areas of pixels), it works well as an example-based explanatory tool, and may still work well to give humans intuitive ideas about patterns to look for.

3.4.2 Saliency Maps

The ideas of saliency maps and occlusion are similar, but the methods are different in implementation. Saliency maps, first demonstrated by Simonyan et al. [2014], are typically implemented as gradient-based tools for visualization of which pixels contributed to increasing the probability of a certain classification. This is algorithmically achieved by first selecting an image to analyze and a class we want to analyze for; the saliency map will show pixel contributions to the given class. The image is sent in a forward pass through the network. The paper then defines a score function $S_c(I)$ as the score given to a class c for a given image I (which essentially is the same as the logit for the class before softmaxing at the final fully connected layer). Though the calculation of the score function is non-linear in a typical CNN, the paper approximates it to a linear function through a first-order Taylor expansion:

$$S_c(I) = w^T I + b \tag{3}$$

This is then differentiated with respect to the input image I_0 that the saliency map is produced for:

$$w = \frac{\delta S_c}{\delta I}|_{I=I_0} \tag{4}$$

Much like when interpreting parameter weights in neural networks in general, the weights w tell us how much each pixel influences the class score; for pixels with large w values, the pixel value only needs to change a little to have an impact on the score. Finally, these values can be visualized on a heat map overlying the original input image to make it human-readable and interpretable.

When compared to 3.4.1, saliency maps are more exploitative of the fact that neural networks by their nature can produce gradients relative to the input data. They produce more local information, meaning feature importance can be traced back even to individual pixels, not only areas as is typically the case for occlusion-based methods. Mathematically speaking, gradients also give a more accurate measure of exactly *how much* influence a pixel has on the class prediction. However, the method described here still does not take full advantage of the fact that neural networks can also produce gradients with respect to their internal components, as we will see in the next method.

3.4.3 Grad-CAM

Gradient-weighted Class Activation Mapping, or Grad-CAM, builds on the same ideas that were introduced in 3.4.2. The main difference is this: where saliency maps calculate direct pixel contributions for a given input image by backpropagating back to the image, Grad-CAM only backpropagates back to the final convolutional layer. In CNNs, lower-level convolutional layers will typically learn to identify primitive concepts, like edges or simple shapes. In higher-level ones, the feature maps from previous layers are combined so that the filters learn to identify higher-level concepts in the form of more complex shapes, like faces or body parts. This means that the layer Grad-CAM backpropagates to will likely be the layer with the most complex feature maps. It is here that Grad-CAM derives its information about feature importance in a given input image.

Algorithmically, Grad-CAM starts similarly to saliency maps by sending an input image in a forward pass through a CNN. We again select a class of interest c, and the forward pass will produce a class score y^c for the image before softmaxing. This score is then differentiated with respect to the activation maps of the last convolutional layer A^k, so we get $\frac{\delta y^c}{\delta A^k}$. Note that A^k are feature maps of height and width dimensions, so to get feature importance from the neurons of each feature map, Grad-CAM performs global average pooling over the gradients for each neuron:

$$\alpha_k^c = \frac{1}{Z}\sum_i\sum_j \frac{\delta y^c}{\delta A_{ij}^k} \tag{5}$$

α_k^c now represents a measure of how much each activation map A^k contributes to producing y^c. Thus, we receive exactly k values α_k^c for a given class c, one per feature map. At this point, the feature maps of the last convolutional layer A^k are averaged over, weighted by their respective α_k^c. Lastly, Grad-CAM applies the ReLU activation function to this average to select only neurons from the feature maps that should be increased to increase y^c. This finally produces the output of Grad-CAM, a coarse heat map for input i with the same height and width dimensions as each of the feature maps:

$$L_{Grad-CAM}^c = ReLU(\sum_k \alpha_k^c A^k) \tag{6}$$

For visualization purposes, this map is typically upscaled to the same size as the input image.

The Grad-CAM algorithm is significant, as it serves as the groundwork for many of the current SOTA algorithms for XAI in the image domain. Selvaraju et al. [2019] also proposed an algorithm to combine Grad-CAM with ideas from saliency maps, known as Guided Grad-CAM, where gradients from input pixels are combined with the aforementioned feature map neurons to capture both important pixels and concepts in the outputted heat map. Several variations deriving from Grad-CAM have been made, e.g. Grad-CAM++ [Chattopadhay et al., 2018], Smooth Grad-CAM++ [Omeiza et al., 2019] and Eigen-CAM [Muhammad and Yeasin, 2020].

4 Improving ML Performance Through XAI

Generally in XAI research, the main motivation behind the development of methods such as the ones discussed in 3 is to explain how a prediction model works and why it makes the predictions it does in a precise and interpretable way. In this section, I want to motivate the use of XAI methods for improving the models themselves and look at the current state of this field of XAI.

Presume that there exists an XAI method capable of explaining a given black-box model's predictions with perfect fidelity and higher interpretability than that of the black-box model itself. This would mean that the method has learned to translate the model's predictions to a more understandable form while keeping all learned information in the model's representation of the dataset it was trained on. This would essentially be equivalent to a mapping from the domain of the outputted prediction by the model, to a constructed domain where no information is lost from the former one, but which is still intrinsically different and perhaps carries different meaning, making it easier to interpret.

Assuming that an XAI method is capable of creating such a mapping with no loss of fidelity in its explanation is of course hypothetical; as Ribeiro et al. [2016] and Nazir et al. [2023] argue, there is typically a trade-off between interpretability and fidelity. Nevertheless, if more interpretability can be gained from a model output at the cost of some fidelity, the resulting explanation from such a mapping could be used to retrain the black-box model, coupling the explanation to the input sample it was generated from. It may provide new information to the model which could help it understand the dataset it is retrained on even better. In principle, this could be thought of simply as a feature transformation. Such transformations are already commonly used in practice to enhance datasets, though typically not through the use of XAI methods.

This argumentation is not to say that such mappings would be the only way of using XAI in the context of improving prediction performance. Nazir et al. [2023] list several cases in which XAI techniques were used to enhance the performance of an ML model in various ways. In the case of deep neural networks, an intuitive way of using XAI methods to improve the network itself is to quantitatively or qualitatively explain the inner workings of the network by applying similar techniques to the ones discussed in 3.4. For instance, Yeom et al. [2021] use LRP, or Layer-wise Relevance Propagation, to perform model pruning, effectivizing the model. Bau et al. [2017] propose a framework for dissecting CNNs, through which they recognize certain characteristics and training patterns of the networks.

Unlike the example-based methods discussed in 3.4, the methodology described in these papers has a global scope. They attempt to describe the entirety of complex models independently of input data. This is ultimately one of the great challenges of XAI in general as well, but it shows that if achieved, good explanations of complex models have a wide range of use cases. In addition to the perhaps more typical applications, such as increasing the reliability and trustworthiness of safety-critical systems, a well-explained model will be easier to modify and improve upon. It would help ML practitioners understand what are the typical components that make the best-performing model architectures good, making model development even more efficient.

References

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 839–847, 2018. doi: 10.1109/WACV.2018.00097.

Arun Das and Paul Rad. Opportunities and challenges in explainable artificial intelligence (xai): A survey, 2020.

Prashant Gohel, Priyanka Singh, and Manoranjan Mohanty. Explainable AI: current status and future directions. *CoRR*, abs/2107.07045, 2021. URL `https://arxiv.org/abs/2107.07045`.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1), 2021. ISSN 1099-4300. doi: 10.3390/e23010018. URL `https://www.mdpi.com/1099-4300/23/1/18`.

Christoph Molnar. *Interpretable Machine Learning*. 2 edition, 2022. URL `https://christophm.github.io/interpretable-ml-book`.

Mohammed Bany Muhammad and Mohammed Yeasin. Eigen-cam: Class activation map using principal components. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2020. doi: 10.1109/IJCNN48605.2020.9206626.

Sajid Nazir, Diane M. Dickson, and Muhammad Usman Akram. Survey of explainable artificial intelligence techniques for biomedical imaging with deep neural networks. *Computers in Biology and Medicine*, 156:106668, 2023. ISSN 0010-4825. doi: https://doi.org/10.1016/j.compbiomed.2023.106668. URL `https://www.sciencedirect.com/science/article/pii/S0010482523001336`.

Daniel Omeiza, Skyler Speakman, Celia Cintas, and Komminist Weldermariam. Smooth grad-cam++: An enhanced inference level visualization technique for deep convolutional neural network models, 2019.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939778. URL `https://doi.org/10.1145/2939672.2939778`.

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. *International Journal of Computer Vision*, 128(2):336–359, October 2019. ISSN 1573-1405. doi: 10.1007/s11263-019-01228-7. URL `http://dx.doi.org/10.1007/s11263-019-01228-7`.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2014.

Mattias Wahde and Marco Virgolin. Daisy: An implementation of five core principles for transparent and accountable conversational ai. *International Journal of Human–Computer Interaction*, 39(9):1856–1873, 2023. doi: 10.1080/10447318.2022.2081762. URL `https://doi.org/10.1080/10447318.2022.2081762`.

Seul-Ki Yeom, Philipp Seegerer, Sebastian Lapuschkin, Alexander Binder, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Pruning by explaining: A novel criterion for deep neural network pruning. *Pattern Recognition*, 115:107899, 2021. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2021.107899. URL `https://www.sciencedirect.com/science/article/pii/S0031320321000868`.

Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In David

Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 818–833, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10590-1.