

Explainable Deep Learning: A Field Guide for the Uninitiated

Gabriëlle Ras

G.RAS@DONDERS.RU.NL

*Donders Institute for Brain, Cognition and Behaviour
Radboud University Nijmegen
6525 HR Nijmegen, the Netherlands*

Ning Xie

XINING@AMAZON.COM

*Amazon
Seattle, Washington, USA*

Marcel van Gerven

M.VANGERVEN@DONDERS.RU.NL

*Donders Institute for Brain, Cognition and Behaviour
Radboud University Nijmegen
6525 HR Nijmegen, the Netherlands*

Derek Doran

DEREK.DORAN@TENET3.COM

*Tenet3, LLC
Dayton, Ohio, USA*

Abstract

Deep neural networks (DNNs) are an indispensable machine learning tool despite the difficulty of diagnosing what aspects of a model’s input drive its decisions. In countless real-world domains, from legislation and law enforcement to healthcare, such diagnosis is essential to ensure that DNN decisions are driven by aspects appropriate in the context of its use. The development of methods and studies enabling the explanation of a DNN’s decisions has thus blossomed into an active and broad area of research. The field’s complexity is exacerbated by competing definitions of what it means “to explain” the actions of a DNN and to evaluate an approach’s “ability to explain”. This article offers a field guide to explore the space of explainable deep learning for those in the AI/ML field who are uninitiated. The field guide: i) Introduces three simple dimensions defining the space of foundational methods that contribute to explainable deep learning, ii) discusses the evaluations for model explanations, iii) places explainability in the context of other related deep learning research areas, and iv) discusses user-oriented explanation design and future directions. We hope the guide is seen as a starting point for those embarking on this research field.

1. Introduction

Artificial intelligence (AI) systems powered by deep neural networks (DNNs) are pervasive across society: they run in our pockets on our cell phones (Georgiev et al., 2017), in cars to help avoid car accidents (Jain et al., 2015), in banks to manage our investments (Chong et al., 2017) and evaluate loans (Pham & Shen, 2017), in hospitals to help doctors diagnose disease symptoms (Nie et al., 2015), at law enforcement agencies to help recover evidence from videos and images to help law enforcement (Goswami et al., 2014), in the military of many countries (Lundén & Koivunen, 2016), and at insurance agencies to evaluate coverage suitability and costs for clients (Dong et al., 2016; Sirignano et al., 2016). However, when

medical treatment is to be assigned or when a significant financial decision must be made, an AI that *suggests* a course of action, rather than to merely *prescribe* one, is desired. For the human ultimately responsible for the action taken, the use of DNNs leaves an important question unanswered: *how can a person that is held accountable for a decision trust a DNN’s recommendation and justify its use?* Trust and justification can hardly be achieved if the user does not have access to a satisfactory explanation for the process that led to the recommendation. Consider, for example, a hypothetical scenario in which a medical system runs a DNN in the backend. Assume that the system makes life-altering predictions about whether or not a patient has a terminal illness. It is desirable if this system could also provide a rationale behind its predictions. Equally important is for the system to give a rationale that both physicians and patients can understand and trust. Trust in a decision is built upon a rationale that is: (i) easily interpretable; (ii) relatable to the user; (iii) connects the decision with contextual information about the choice or to the user’s prior experiences; and (iv) reflects the intermediate thinking of the user in reaching a decision. Given the qualitative nature of these characteristics, it may come as no surprise that there is great diversity in the definitions, approaches, and techniques used by researchers to provide a rationale for the decisions of a DNN. This diversity is further compounded by the fact that the form of a rationale often conforms to a researcher’s personal notion of what constitutes an “explanation”. For a newcomer, whether a seasoned researcher or a student in disciplines that DNNs are impacting, jumping into the field is a daunting task.

This article offers a starting point for researchers and practitioners who are embarking on the field of explainable deep learning. This field guide is designed to help newcomers understand:

- A set of **dimensions** characterizing the space of foundational work in explainable deep learning and a description of such methods. This space summarizes the core aspects of explainable DNN techniques that a majority of present work is inspired by or built from (**Section 2**).
- Methods for **evaluating** explanation methods (**Section 3**).
- Complementary **research topics** that are aligned with explainability, such as how DNNs learn to generalize or approaches to reduce a DNN’s sensitivity to particular input features. The topics are indirectly associated with explainability in the sense that they investigate *how* a DNN learns or performs inference (**Section 4**).
- The **considerations** of a designer developing an explainable DNN system (**Section 5**).
- **Future directions** in explainability research (**Section 6**).

Our taxonomy of explainable DNN techniques clarifies the technical ideas underpinning most modern explainable deep learning techniques. The discussion of fundamental explainable deep learning methods, emblematic of each framework dimension, provides further context for the modern work that builds on or takes inspiration from them. Following the taxonomy is a brief discussion on the evaluations of model explanations. Complementary DNN topics are reviewed, and the relationships between explainable DNNs and other related research areas are developed. The field guide then turns to essential considerations that need

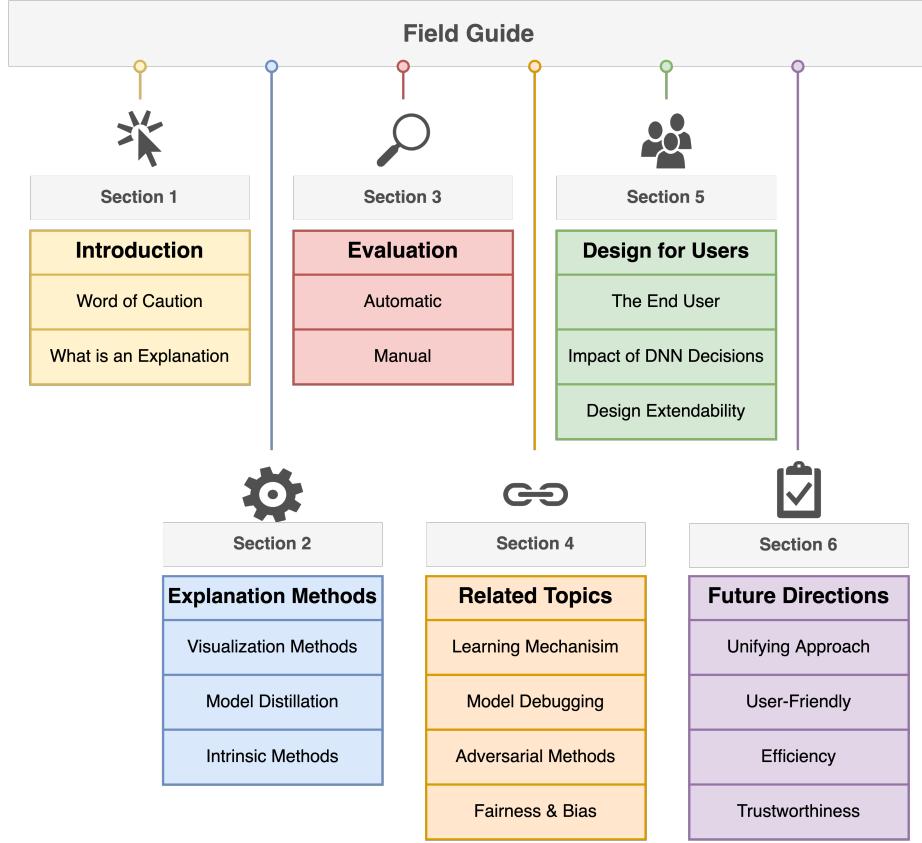


Figure 1: Outline of the field guide.

to be made when building an explainable DNN system in practice, considering the end-user. Finally, the overview of our current limitations and seldom-looked at aspects of explainable deep learning suggest new research directions. This information captures what a newcomer needs to know to successfully navigate the current research literature on explainable deep learning and identify new research problems.

There are many reviews on the topic of model explainability. Most of them focus on explanations of general artificial intelligence methods (Arrieta et al., 2020; Carvalho et al., 2019; Mueller et al., 2019; Tjoa & Guan, 2020; Gilpin et al., 2018; Adadi & Berrada, 2018; Miller, 2019; Guidotti et al., 2018; Lipton, 2018; Liu et al., 2017; Došilović et al., 2018; Doshi-Velez & Kim, 2017; Molnar, 2020; Carvalho et al., 2019), and some on deep learning (Ras et al., 2018; Montavon et al., 2018; Zhang & Zhu, 2018; Samek et al., 2017; Erhan et al., 2010). The unique **contributions** of this field guide are as follows. First, it targets explanations for deep learning systems, while existing reviews focus on explanations of general artificial intelligence methods or have a narrower focus on particular types of deep learning architectures. Second, it is designed to support researchers uninitiated in explainable deep learning and, hopefully, lower the bar to enter this field. Third, it introduces a novel categorization scheme to systematically organize numerous explanation methods, with an eye towards simplicity and focus on the field’s foundations. Finally, the review

connects related fields to explainable deep learning to better understand how they contribute to existing work to improve DNN transparency, robustness, and reliability.

1.1 A Word of Caution

Although this paper focuses on explaining DNNs, it does not mean that DNNs are the only problem-solving tools in a machine learning toolbox. DNNs have significant potential for misuse when applied prematurely or incorrectly. Extreme caution must be taken, by all parties involved, to ensure that the DNN technology and derivatives are properly tested before production and commercial use. It is also recommended to review the perceived need for DNNs and consider if other algorithms can serve the same purpose (Rudin, 2019). One example of this is the wrongful use of facial recognition technology by federal law enforcement agencies. Studies have shown that facial recognition technology is significantly less accurate on non-white, non-male persons (Buolamwini & Gebru, 2018; Garvie, 2016), yet the immature technology was used in the real world with negative outcomes.

On the other hand, DNN applications in domains such as medicine (Rajpurkar et al., 2018; Esteva et al., 2017) or applications that can benefit climate change (Rolnick et al., 2019) have been meaningful and beneficial to society as a whole. While the paper looks at explainability through the lens of the DNNs, the need for explanations extends to the whole of scientific reasoning.

1.2 But What is an Explanation?

Defining *explanation* is a philosophical activity that does not align with the goal of this paper. Instead, we describe examples of what explanations can look like in the context of deep learning. In its most general form, an explanation is any information that can help the user understand and communicate why a model exhibits some pattern of decision-making and how individual decisions come about.

The goal of any explanation can roughly fall into one of the following two categories: (i) *explanations that give insight into model training and generalization*. These explanations give a practitioner additional information that can be used to make decisions about the components in the model training and validation process, e.g., the number of labeled data, value of the hyperparameters, and model choice. The other category is (ii) *explanations that give insight into model predictions*. Most explanations fall into this category and help practitioners explain why the model made a particular prediction, usually in terms of the model input. These explanations can be used to communicate to others (potentially non-experts) a model decision. Many individual predictions can be analyzed to reveal patterns in overall model prediction behavior. This category of explanations can further be broken down in more specific categories such as *counterfactual explanations* (Verma et al., 2020) and *contrastive explanations* (Miller, 2021).

Most explanations bear a strong resemblance to the data type that was used to train the DNN. If the datatype is an image, for example, the explanation can be a saliency or heatmap. A saliency map depicts regions in the image that the explanation method determined was important for the network’s prediction. If the datatype is text-based, the explanation can look like highlighted words in the text. If the data is composed of attributes, i.e., data that can be represented as a table, the explanation can be a set of rules that describe which

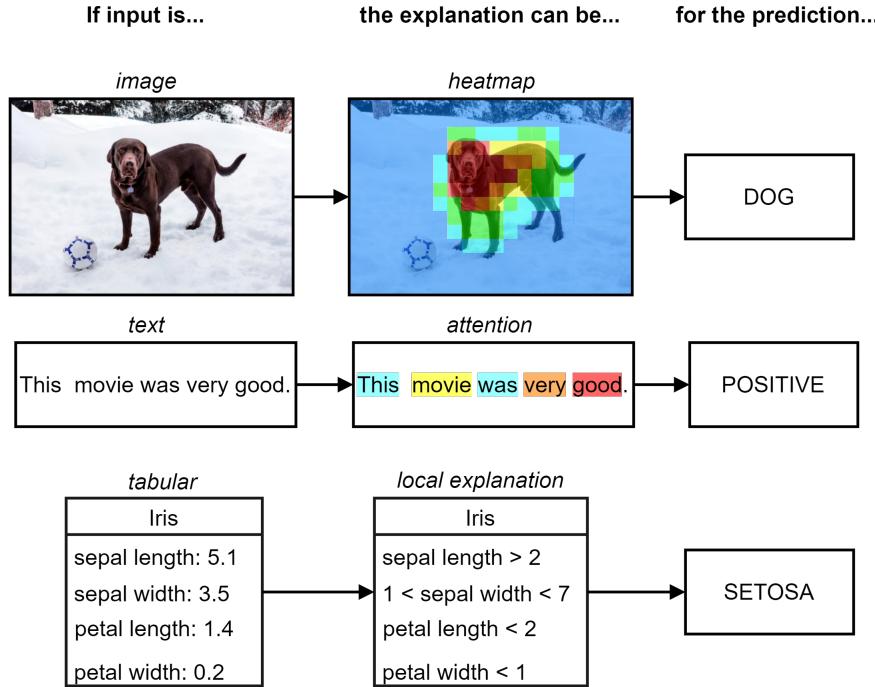


Figure 2: Examples of what explanations can look like in practice. The explanation depends on the type of data used and the method used to create the explanation. In general, an explanation is any information that aids the user in achieving insight into model training, generalization, or rationale behind the model prediction.

combinations of different attribute values lead to which predictions. The illustrations in Figure 2 reflect different explanation representations based on data type. It is useful to keep in mind that while heatmaps are one of the most common and natural ways to present a model explanation, it is also subject to interpretation by the practitioner. Practitioners can interpret the explanation differently, especially when the explanation method is unknown.

2. Methods for Explaining DNNs

There are countless surveys on explainable AI (Arrieta et al., 2020; Carvalho et al., 2019; Mueller et al., 2019; Tjoa & Guan, 2020; Gilpin et al., 2018; Adadi & Berrada, 2018; Miller, 2019; Guidotti et al., 2018; Lipton, 2018; Liu et al., 2017; Došilović et al., 2018; Doshi-Velez & Kim, 2017) and explainable deep learning (Ras et al., 2018; Montavon et al., 2018; Zhang & Zhu, 2018; Samek et al., 2017; Erhan et al., 2010). The surveys cover a large body of work that may prove hard to navigate and synthesize into a broad view of the field. Instead, this article investigates a simple space of **foundational** explainable DNN methods. We say a method is foundational if it is often used in practice or introduces a concept that modern work builds upon. Understanding this smaller space of foundational methods will support a reader as they study modern approaches.

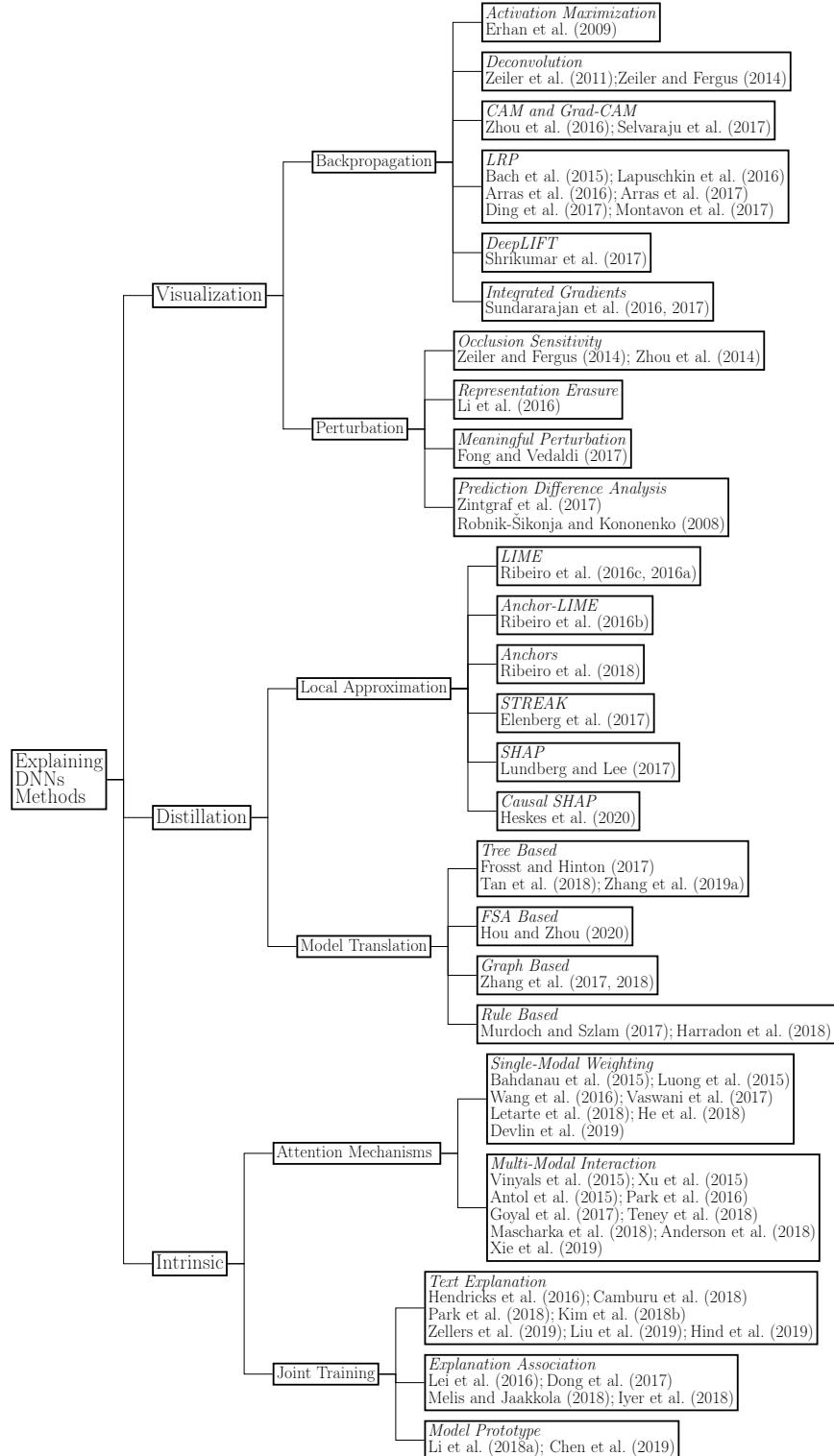


Figure 3: Methods for explaining DNNs.

Since different users at different stages of the software pipeline have different requirements, it is only possible to represent relative advantages given the explainability goal that needs to be achieved. The users that we will take into account for this discussion are the *expert users* as described in Ras et al. (2018).

We present a simple three-dimensional space encompassing:

- **Visualization methods:** Visualization methods express an explanation by highlighting, through a scientific visualization, characteristics of an input that strongly influence the output of a DNN.
- **Model distillation:** Model distillation develops a separate, “white-box” machine learning model that is trained to mimic the input-output behavior of the DNN. The white-box model, which is inherently explainable, is meant to identify the decision rules or input features influencing DNN outputs.
- **Intrinsic methods:** Intrinsic methods are DNNs that have been specifically created to render an explanation along with its output. As a consequence of its design, intrinsically explainable deep networks can jointly optimize model performance and the form of the explanations produced.

2.1 Visualization Methods

Visualization Methods	Summary	References
Backpropagation-based	Visualize feature relevance from volume of gradient passed through layers during network training.	Erhan et al. (2009), Zeiler et al. (2011), Zeiler and Fergus (2014), Zhou et al. (2016), Selvaraju et al. (2017), Bach et al. (2015), Lapuschkin et al. (2016), Arras et al. (2016, 2017), Ding et al. (2017), Montavon et al. (2017), Shrikumar et al. (2017), Sundararajan et al. (2017, 2016)
Perturbation-based	Visualize feature relevance by comparing network output of an input and a modified copy of the input.	Zeiler and Fergus (2014), Zhou et al. (2014), Li et al. (2016), Fong and Vedaldi (2017), Robnik-Šikonja and Kononenko (2008), Zintgraf et al. (2017)

Table 1: Visualization methods.

Visualization methods associate the degree to which a DNN considers input features to a decision. This association is often referred to as *attribution*. A common explanatory form of visualization methods is *saliency maps* or *heatmaps*, where oftentimes a transparent colored heatmap is overlaid on the original input image. These maps identify input features that are most salient, in the sense that they cause a maximum response or stimulation influencing the model’s output (Yosinski et al., 2015; Ozbulak, 2019; Olah et al., 2017, 2018; Carter et al., 2019). We break down visualization methods into two types, namely *backpropagation* and *perturbation*-based visualization. The types are summarized in Table 1 and will be discussed further below.

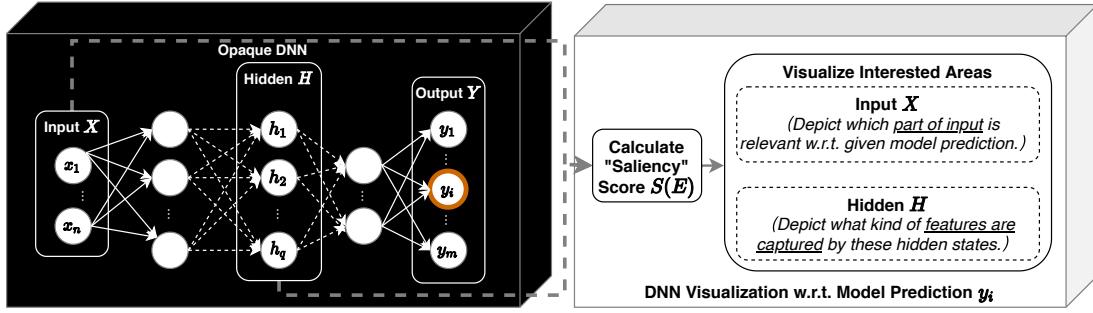


Figure 4: Visualization Methods. The to-be-visualized element E can be from either the model input X or hidden states H . Visualization is based on the calculated saliency score $S(E)$, which varies along with different visualization methods.

2.1.1 BACKPROPAGATION-BASED METHODS

Backpropagation-based methods identify the saliency of input features based on some evaluation of gradient signals passed from output to input during network training. A baseline gradient-based approach visualizes the partial derivative of the network output with respect to each input feature scaled by its value (Simonyan et al., 2013; Springenberg et al., 2014), thus quantifying the sensitivity of the network’s output with respect to input features. In a scene recognition task, for example, a high relevance score for pixels representing a bed in a CNN that decides the image is of class “bedroom” may suggest that the decision made by the CNN is highly sensitive to the presence of the bed in the image. Other gradient-based methods may evaluate this sensitivity with respect to the output, but from different collections of feature maps at intermediate CNN network layers (Zeiler & Fergus, 2014; Bach et al., 2015; Montavon et al., 2017; Shrikumar et al., 2017).

Activation Maximization. One of the earliest works on visualization in deep architectures is proposed by Erhan et al. (2009). This seminal study introduces the activation maximization method to visualize important features in any layer of a deep architecture by optimizing the input \mathbf{X} such that the activation a of the chosen unit i in a layer j is maximized:

$$\arg \max_{\mathbf{X}} a_{i,j}(\mathbf{X}, \boldsymbol{\theta}) \quad (1)$$

Parameters $\boldsymbol{\theta}$ of a trained network are kept fixed during activation maximization. The optimal \mathbf{X} is found by computing the gradient of $a_{i,j}(\mathbf{X}, \boldsymbol{\theta})$ and updating \mathbf{X} in the direction of the gradient. The practitioner decides the values of the hyperparameters for this procedure, i.e., the learning rate and how many iterations to run. The optimized \mathbf{X} will be a representation, in the input space, of the features that maximize the activation of a specific unit, or if the practitioner chooses so, multiple units in a specific network layer.

By visualizing the internal representations, the practitioner can check if concepts learned by the model are human interpretable. The quality of the concepts can be used as an indication of model generalizability and determine if, for instance, additional labeled data is needed to train the model. Activation maximization can give insight into the model training

and generalization process but does not lend itself to explaining individual model predictions.

Deconvolution. Deconvolution was originally introduced as an algorithm to learn image features in an unsupervised manner (Zeiler et al., 2011). However, the method gained popularity because of its applications in visualizing higher layer features in the input space (Zeiler & Fergus, 2014), i.e., visualizing higher layer features in terms of the input. Deconvolution assumes that the model being explained is a neural network consisting of multiple convolutional layers. We will refer to this model as CNN. The consecutive layers of this network consist of a convolution of the previous layer’s output (or the input image in the case of the first convolutional layer) with a set of learned convolutional filters, followed by the application of the rectified linear function (ReLU) $\text{ReLU}(\mathbf{A}) = \max(\mathbf{A}, 0)$ on the output of the aforementioned convolution

$$\mathbf{A}^\ell, s^\ell = \text{maxpool} \left(\text{ReLU} \left(\mathbf{A}^{\ell-1} * \mathbf{K}^\ell + \mathbf{b}^\ell \right) \right) \quad (2)$$

where ℓ indicates the respective layer, \mathbf{A}^ℓ is the output of the previous layer, \mathbf{K} is the learned filter, and \mathbf{b} is the bias. If the outputs from the ReLU are passed through a local max-pooling function, it additionally stores the output s^ℓ containing the indices of the maximum values for a later unpooling operation. In the original paper, the set of s^ℓ ’s are referred to as *switches*. A deconvolutional neural network, referred to as DeCNN, consists of the inverse operations of the original convolutional network CNN. DeCNN takes the output of CNN as its input. In other words, DeCNN runs the CNN in reverse, from top-down. This is why the deconvolution method is classified as a backpropagation method. The convolutional layers in CNN are replaced with *deconvolutions* and the max-pooling layers are replaced with *unpooling* layers. A deconvolution is also called a *transposed convolution*, meaning that the values of \mathbf{K}^ℓ are transposed and then copied to the deconvolution filters $\mathbf{K}^{\ell T}$. If the CNN included max-pooling layers, they are replaced with unpooling layers which approximately upscales the feature map, retaining only the maximum values. This is done by retrieving the indices stored in s^ℓ at which the maximum values were located when the max-pooling was originally applied in CNN.

As an example let us see the calculations involved in deconvolving Equation 2:

$$\mathbf{A}^{\ell-1} = \text{unpool} \left(\text{ReLU} \left((\mathbf{A}^\ell - \mathbf{b}^\ell) * \mathbf{K}^{\ell T} \right), s^\ell \right) \quad (3)$$

Using Equation 3 one or multiple learned filters \mathbf{K} in any layer of the network can be visualized by reverse propagating the values of \mathbf{K} all the way back to the input space. Finally, this study also describes how the visualizations can be used for architecture selection.

Practitioners can use this method to visualize how much information from the original input the extracted features retain, and gain insight on how information is extracted from data at different network layers. From this insight actions can be taken to improve the model training process. Deconvolution does not lend itself to explaining single model predictions.

CAM and Grad-CAM. Zhou et al. (2016) describes a visualization method for creating class activation maps (CAM) using global average pooling (GAP) in CNNs. Lin et al. (2013) proposes the idea to apply a global average pooling on the activation maps of the last convolutional layer, right before the fully connected (FC) output layer. This results in

the following configuration at the end of the CNN: $\text{GAP}(\text{Conv}) \rightarrow \text{FC} \rightarrow \text{softmax}$. The FC layer has C nodes, one for each class. The CAM method combines the activations \mathbf{A} from Conv, containing K convolutional filters, and weights $w_{k,c}$ from FC, where the (k, c) pair indicates the specific weighted connection from Conv to FC, to create relevance score map:

$$\text{map}_c = \sum_k^K w_{k,c} \mathbf{A}_k \quad (4)$$

The map is then upsampled to the size of the input image and overlaid on the input image, very similar to a heat map, resulting in the class activation map. Each class has a unique CAM, indicating the image regions important to network prediction for that class. CAM can only be applied on CNNs that employ the $\text{GAP}(\text{Conv}) \rightarrow \text{FC} \rightarrow \text{softmax}$ configuration. Gradient-weighted Class Activation Map (**Grad-CAM**) (Selvaraju et al., 2017) is a generalization CAM using the gradients of the network output with respect to the last convolutional layer to achieve the class activation map. This allows Grad-CAM to be applicable to a broader range of CNNs compared to CAM, only requiring that the final activation function used for network prediction to be a differentiable function, e.g., softmax. For each feature map \mathbf{A}_k in the final convolutional layer of the network, a gradient of the score y_c (the value before softmax, also known as *logit*) of class c with respect to every node in \mathbf{A}_k is computed and averaged to get an importance score $\alpha_{k,c}$ for feature map \mathbf{A}_k :

$$\alpha_{k,c} = \frac{1}{m \cdot n} \sum_{i=1}^m \sum_{j=1}^n \frac{\partial y_c}{\partial \mathbf{A}_{k,i,j}} \quad (5)$$

where $\mathbf{A}_{k,i,j}$ is a neuron positioned at (i, j) in the $m \times n$ feature map \mathbf{A}_k . Grad-CAM linearly combines the importance scores of each feature map and passes them through a ReLU to obtain an $m \times n$ -dimensional relevance score map

$$\text{map}_c = \text{ReLU} \left(\sum_k^K \alpha_{k,c} \mathbf{A}_k \right) \quad (6)$$

The relevance score map is then upsampled via bilinear interpolation to be of the same dimension as the input image to produce the class activation map. Figure 5 shows a visual representation of the grad-CAM method and an example of what a grad-CAM heatmap looks like for the prediction “cat”.

Practitioners can use the CAM family of methods to determine, given an input and a class, what is the information in the input that gives evidence for that class. Based on this information the practitioner can determine to what extent model predictions can be interpreted and assess for which classes consistent model predictions can be expected. For example, if we have two models where both have the same accuracy score, a model that produces heatmaps consistent with human experience is often considered more trustworthy compared to one where the heatmaps correspond poorly to human experience. Practitioners can also use the CAM family of methods to determine if there is an unfavorable class bias that the model is picking up on, e.g., skin color. The reader should note that only positive attribution can be computed with this method due to the ReLU function in Equation 6.

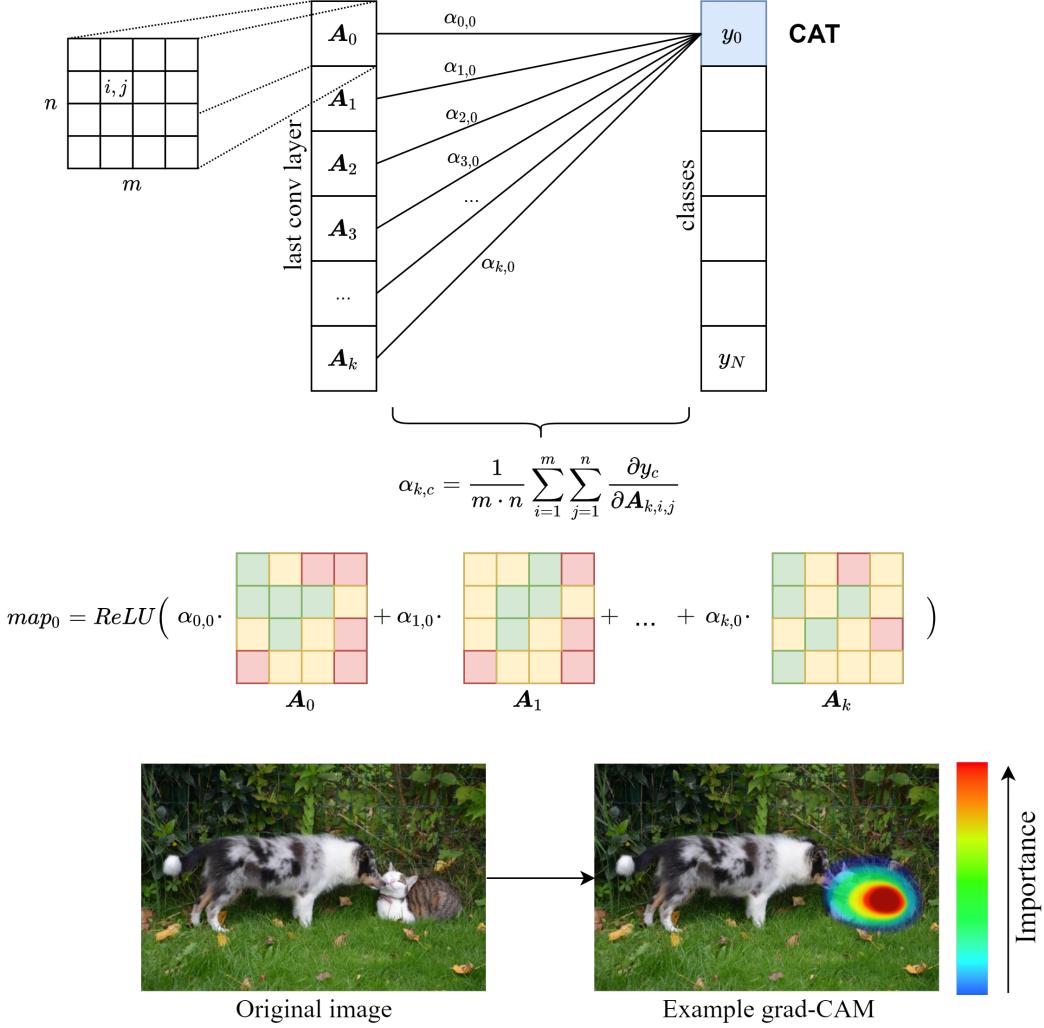


Figure 5: Visual explanation of how grad-CAM works. **Top:** Visualization of Equation 5 for calculating the importance scores $\alpha_{i,j}$ for each feature map \mathbf{A}_k . **Middle:** The heatmap for a specific class is computed by multiplying the importance score with each feature map and taking the sum. Afterwards the heatmap is upsampled and overlaid on the original image. **Bottom:** Heatmap for the prediction ‘cat’.

Layer-Wise Relevance Propagation. LRP methods create a saliency map that, rather than measuring sensitivity, represents the relevance of each input feature to the output of the network (Bach et al., 2015; Lapuschkin et al., 2016; Arras et al., 2016, 2017; Ding et al., 2017; Montavon et al., 2017). While *sensitivity* measures the change in response in the network’s output as a result of changing attributes in the input (Kindermans et al., 2019), *relevance* measures the strength of the connection between the input features to network output, without making any changes to the input or the components of the network.

LRP methods decompose the output value $f(\mathbf{x})$ of a deep network f across input features $\mathbf{x} = (x_1, x_2, \dots, x_N)$, such that $f(\mathbf{x}) = \sum_i r_i$ where r_i is the relevance score of feature x_i .

Perhaps the most generic type of LRP is called Deep Taylor Decomposition (Montavon et al., 2017). The method is based on the fact that f is differentiable and hence can be approximated by a Taylor expansion of f at some root $\hat{\mathbf{x}}$ for which $f(\hat{\mathbf{x}}) = 0$:

$$\begin{aligned} f(\mathbf{x}) &= f(\hat{\mathbf{x}}) + \nabla_{\hat{\mathbf{x}}} f \cdot (\mathbf{x} - \hat{\mathbf{x}}) + \epsilon \\ &= \sum_i^N \frac{\partial f}{\partial x_i}(\hat{x}_i) \cdot (x_i - \hat{x}_i) + \epsilon \end{aligned} \quad (7)$$

where ϵ encapsulates all second order and higher terms in the Taylor expansion. A good root point is one that is as minimally different from \mathbf{x} and causes the function $f(\mathbf{x})$ to output a different prediction. The relevance score for inputs can then be seen as the terms inside of the summation:

$$r_i = \frac{\partial f}{\partial x_i}(\hat{x}_i) \cdot (x_i - \hat{x}_i) \quad (8)$$

To extend this idea to a DNN, the deep Taylor decomposition algorithm considers a conservative decomposition of relevance scores across layers of the network, starting from the output, through each hidden layer, back to the input. Thus, the method requires that the relevance score of a node i at layer l , denoted r_i^l be decomposable into

$$r_i^l = \sum_j^M r_{i,j}^l \quad (9)$$

where the summation is taken over all M nodes in layer $l+1$ that node i in layer l connects or contributes to. This indicates that the relevance score of the later layers can be backpropagated to generate the relevance score of former layers. The relevance score with respect to the input space can thus be calculated by conducting this decomposition rule layer by layer. Further details can be found in the original paper (Montavon et al., 2017).

In practice, the results of applying LRP are similar to the results of the CAM family of methods: given an input and a prediction, both methods tell the practitioner the regions in the input that are most relevant for the prediction. LRP allows the heatmap to display negative attributions in addition to positive ones. Practitioners can use the information to assess and further investigate things like model bias, prediction consistency and model trust. However, with LRP the practitioner has to additionally supply the method with a reference input (root) $\hat{\mathbf{x}}$, which in some cases can either be unsolvable or expensive to compute. It is worth noting that the visualization results of LRP rely on the root choices: depending on the input space restrictions, a different root can be chosen, and for different roots chosen, the relevance propagation rule varies, which ultimately yields different appearances of the heatmap. Compared to CAM methods, the LRP heatmap could be of higher quality and more precise. This is because LRP assigns each individual pixel a relevance score, as opposed to CAM, which looks at activation maps in the final layer. The final CAM heatmap will be an upsampled image indicating an approximate relevant region.

DeepLIFT. Deep Learning Important FeaTures (DeepLIFT) is an important approach based on backpropagation by Shrikumar et al. (2017). It assigns relevance scores to input features based on the difference between an input \mathbf{x} and a “reference” input \mathbf{x}' . The reference should be chosen according to the problem at hand and can be found by answering the question “*What am I interested in measuring differences against?*”. In an example using MNIST the reference may be an input of all zeros as this is the background value in the images. Define $\Delta t = f(\mathbf{x}) - f(\mathbf{x}')$ as the difference-from-reference of an interested neuron output of the network between \mathbf{x} and reference \mathbf{x}' , and $\Delta \mathbf{x} = \mathbf{x} - \mathbf{x}'$ as the difference between \mathbf{x} and \mathbf{x}' . DeepLIFT assigns a relevance score $R_{\Delta x_i \Delta t}$ for input feature x_i :

$$\Delta t = \sum_{i=1}^N R_{\Delta x_i \Delta t} \quad (10)$$

where N is the number of input neurons that are necessary to compute t . In this formulation, $R_{\Delta x_i \Delta t}$ can be thought of as a weight denoting how much influence Δx_i had on Δt . According to Equation 10 the sum of the all weights is equal to the difference-from-reference output Δt . The relevance score can be calculated via the *Linear* rule, *Rescale* rule, or *RevealCancel* rule, as elaborated in their study. A multiplier $m_{\Delta \mathbf{x} \Delta t}$ is defined as

$$m_{\Delta \mathbf{x} \Delta t} = \frac{R_{\Delta \mathbf{x} \Delta t}}{\Delta \mathbf{x}} \quad (11)$$

indicating the relevance of $\Delta \mathbf{x}$ with respect to Δt , averaged by $\Delta \mathbf{x}$. Given a hidden layer ℓ of nodes $\mathbf{a}^\ell = (a_1^\ell, a_2^\ell, \dots, a_K^\ell)$, whose upstream connections are the input nodes $\mathbf{x} = (x_1, x_2, \dots, x_N)$, and a downstream target node is t , the DeepLIFT paper proves the effectiveness of the “chain rule” as illustrated below:

$$m_{\Delta x_i \Delta t} = \sum_{j=1}^K m_{\Delta x_i \Delta a_j^\ell} m_{\Delta a_j^\ell \Delta t} \quad (12)$$

This “chain rule” allows for layer-by-layer computation of the relevance scores of each hidden layer node via backpropagation. The DeepLIFT paper and appendix specify particular rules for computing $m_{\Delta x_i \Delta a_j^\ell}$ based on the architecture of the hidden layer \mathbf{a}^ℓ .

DeepLIFT resembles LRP because the practitioner has to choose a reference point and because the algorithm assigns pixel-wise relevance scores. The heatmap results of DeepLIFT explain the difference in prediction between the reference image and the original prediction. Heatmap interpretation depends on which reference image is chosen. By being creative with the choice of the reference image, the practitioner can use DeepLIFT to probe the network in different ways. Heatmaps obtained with DeepLIFT can also display negative attribution in addition to positive attribution, unlike the CAM family.

Integrated Gradients. Integrated gradients (Sundararajan et al., 2017) is an “axiomatic attribution” map that satisfies two axioms for input feature relevance scoring on a network f . The first axiom is *sensitivity*: compared to some baseline input \mathbf{x}' , when input \mathbf{x} differs from \mathbf{x}' along feature x_i and $f(\mathbf{x}) \neq f(\mathbf{x}')$, then x_i should have a non-zero relevance score. The second axiom is *implementation invariance*: for two networks f_1 and f_2 whose outputs are equal for all possible inputs, the relevance score for every input feature x_i should be identical

over f_1 and f_2 . The break of the second axiom may potentially result in the sensitivity of relevance scores on irrelevant aspects of a model.

Given a deep network f whose codomain is $[0, 1]$, an input \mathbf{x} , and a baseline input \mathbf{x}' , the relevance of feature x_i of input \mathbf{x} over f is taken as the integral of the gradients of f along the straight line path from \mathbf{x}' to \mathbf{x} :

$$IG_i(\mathbf{x}) = (x_i - x'_i) \int_0^1 \frac{\partial f(\mathbf{x}' + \alpha(\mathbf{x} - \mathbf{x}'))}{\partial x_i} d\alpha \quad (13)$$

where α is associated with the path from \mathbf{x}' to \mathbf{x} , and is smoothly distributed in range $[0, 1]$. An interpretation of IG_i is the cumulative sensitivity of f to changes in feature i in all inputs on a straight line between \mathbf{x}' to \mathbf{x} along direction i . Intuitively, x_i should have increasing relevance if gradients are large between a “neutral” baseline point \mathbf{x}' and \mathbf{x} along the direction of x_i . In practice, the integral can be approximated by a Riemann summation:

$$IG_i(\mathbf{x}) \approx (x_i - x'_i) \sum_{k=1}^M \frac{\partial F(\mathbf{x}' + \frac{k}{M}(\mathbf{x} - \mathbf{x}'))}{\partial x_i} \frac{1}{M} \quad (14)$$

where M is the number of steps in the approximation. In the original paper, the authors propose setting M between 20 and 300 steps.

IG is similar to LRP and DeepLIFT: the practitioner needs to supply a reference (baseline) image, and the algorithm assigns pixel-wise relevance scores. For image input problems, the baseline image is set to a black image, while for text input, the baseline input can be a zero embedding vector. IG targets the sensitivity axiom not considered by gradient-based attribution methods such as Simonyan et al. (2013), Springenberg et al. (2014) and the implementation invariance axiom not considered by methods like DeepLIFT and LRP.

2.1.2 PERTURBATION-BASED METHODS

Perturbation-based methods compute input feature relevance by altering or removing the input feature and comparing the difference in network output between the original and altered one. Perturbation methods can compute the marginal relevance of each feature with respect to how a network responds to a particular input.

Occlusion Sensitivity. The approach proposed by Zeiler and Fergus (2014), applicable for spatial data, sweeps a ‘grey patch’ that occludes spatial values (i.e., pixels) over the input and sees how the model prediction varies when the patch covers different regions in the input. The reasoning behind this approach is that the model’s performance decreases when the model does not have access to the relevant information. Thus, the more the model performance decreases, the more relevant the occluded region is assumed to be. When a significant portion of the image is swept, the information can be used to create a sensitivity heatmap. In Figure 6 an example is given where a gray patch is swept across an image of a hummingbird. A variant of this method is implemented in (Zhou et al., 2014), where small gray squares are used to occlude image patches in a dense grid. All other methods in this category work similarly. The methods vary in the information that the patch provides or removes, the size of the patch, and how the patches are sampled.

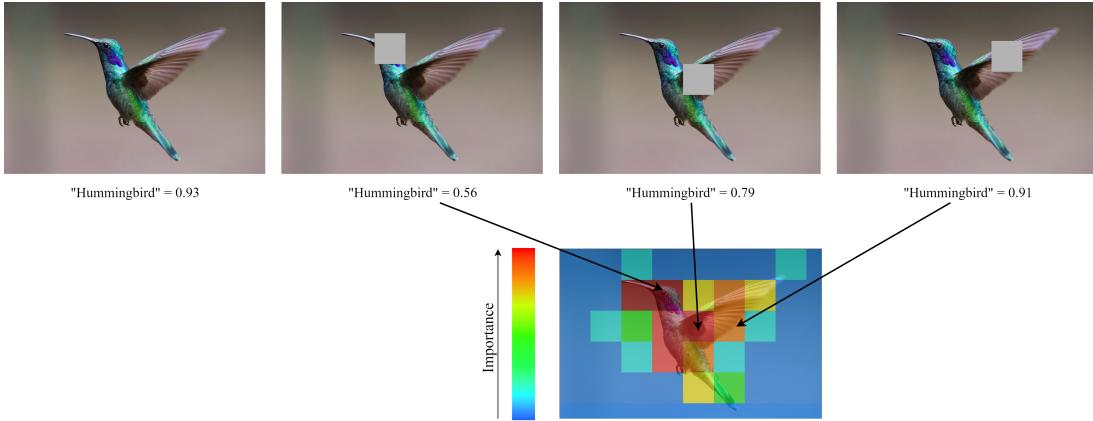


Figure 6: An illustration of how perturbation methods work on images.

The practitioner can use this method to measure how sensitive a model is to a particular part of the removed input. This information can serve as a perfunctory indication of how regions in the input are correlated with model predictions. Since this method does not make use of model internals, it can be used on any DNN. It is worth noting that the higher the desired resolution of heatmaps, the smaller the patch should be, thus the longer it takes to compute the heatmap. Certain features in the input might co-occur, and the joint presence of these features is important. However, occlusion sensitivity is unable to handle this because only one region at a time is occluded.

Representation Erasure. Li et al. (2016) gives an example of a perturbation-based method for natural language input. To measure the effectiveness of each input word or each dimension of intermediate hidden activations, the method erases the information by deleting a word or setting a dimension to zero and observes the influences on model predictions correspondingly. Reinforcement learning (RL) is adopted to evaluate the influence of multiple words or phrases combined by finding the minimum changes in the text that causes a flipping of a neural network’s decision.

Practitioners can use representation erasure to achieve the same goals as occlusion sensitivity: to measure how sensitive a model is to a particular part of the removed input. Even though this method focuses on natural language explanations, it can be modified and applied to other types of problems. Compared to occlusion sensitivity, this method has a couple of benefits. First, representation erasure can handle combinations of erasures. It becomes possible to take into account co-occurring input features. Second, representation erasure is efficient because RL is used to find the minimum change that causes the model’s prediction to change. In contrast, occlusion sensitivity requires the practitioner to sweep the entire image in a brute force manner.

Meaningful Perturbation. Fong and Vedaldi (2017) defines an explanation as a meta-predictor, which is a rule that predicts the output of a black box f to certain inputs x . For example, the explanation for a classifier that identifies a bird in an image can be defined as

$$B(\mathbf{x}; f) = \{\mathbf{x} \in \mathbf{X}_c \Leftrightarrow f(\mathbf{x}) = +1\} \quad (15)$$

where $f(\mathbf{x}) = +1$ means a bird is present and \mathbf{X}_c is the set of all images that the DNN predicts a bird exists. Given a specific image \mathbf{x}^0 and a DNN f , the visualization is generated via perturbation to identify sensitive areas of \mathbf{x}^0 with respect to the output $f(\mathbf{x}^0)$ formulated as a local explanation (“local” to \mathbf{x}^0) by the author. The author defines three kinds of perturbations to delete information from image, i) *constant*, replacing region with a constant value ii) *noise*, adding noise to the region, and iii) *blur*, blurring the region area, and generating explainable visualization respectively.

A practitioner can use meaningful perturbation to find regions in the input that the model’s output is sensitive to, similar to the previous perturbation-based methods. In contrast to said methods, information is never entirely removed from the image, and the amount of perturbation is kept to a minimum. Because of this, the resulting heatmap is more concentrated around the region of interest, and the number of spurious areas is far less compared to the other methods. From a practitioner’s perspective, this makes for more easily interpretable heatmaps.

Prediction Difference Analysis. Zintgraf et al. (2017) proposes a rigorous approach to delete information from an input and measure its influence accordingly. The method is based on (Robnik-Šikonja & Kononenko, 2008), which evaluates the effect of feature x_i with respect to class c by calculating the prediction difference between $p(c | \mathbf{x}_{-i})$ and $p(c | \mathbf{x})$ using the marginal probability

$$p(c | \mathbf{x}_{-i}) = \sum_{x_i} p(x_i | \mathbf{x}_{-i}) p(c | \mathbf{x}_{-i}, x_i) \quad (16)$$

where \mathbf{x} denotes all input features, \mathbf{x}_{-i} denotes all features except x_i , and the sum iterates over all possible values of x_i . The prediction difference, also called *relevance value* in the paper, is then calculated by

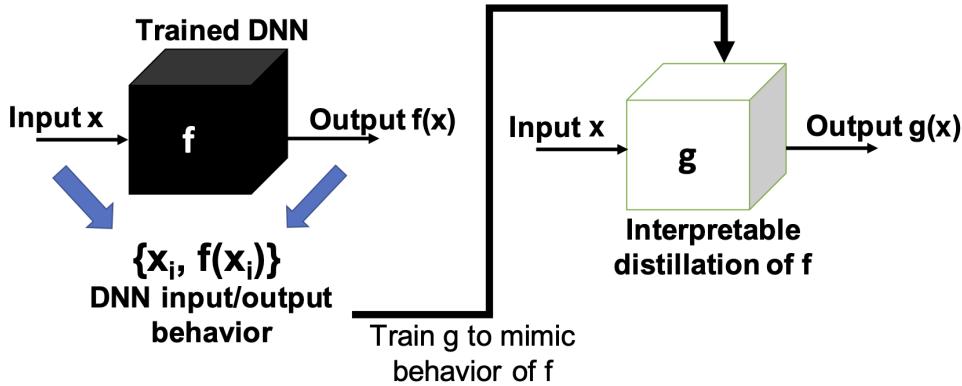
$$\text{Diff}_i(c | \mathbf{x}) = \log_2(\text{odds}(c | \mathbf{x})) - \log_2(\text{odds}(c | \mathbf{x}_{-i})) \quad (17)$$

where $\text{odds}(c | \mathbf{x}) = \frac{p(c | \mathbf{x})}{1 - p(c | \mathbf{x})}$. The magnitude of $\text{Diff}_i(c | \mathbf{x})$ measures the importance of feature x_i . $\text{Diff}_i(c | \mathbf{x})$ measures the influence direction of feature x_i , where a positive value means *for* decision c and a negative value means *against* decision c . Compared to Robnik-Šikonja and Kononenko (2008), Zintgraf et al. improves prediction difference analysis in three ways: by i) sampling patches instead of pixels given the high pixel dependency nature of images; ii) removing patches instead of individual pixels to measure the prediction influence given the robustness nature of neural networks on individual pixels; and iii) adapting the method to measure the effect of intermediate layers by changing the activations of a given intermediate layer and evaluating the influence on down-streaming layers. The heatmaps produced contain both evidence for and against the predicted class.

The practitioner can use this method to gain insight into which regions in the input the model is sensitive to and how perturbations in various model regions affect the output. Compared to the other perturbation-based methods, prediction difference analysis applies a conditional sampling algorithm to determine which regions are perturbed with Gaussian

Model Distillation	Comments	References
Local Approximation	Learns a simple model whose input/output behavior mimics that of a DNN for a small subset of input data.	Ribeiro et al. (2016c, 2016a, 2016b, 2018), Elenberg et al. (2017)
Model Translation	Train an alternative smaller model that mimics the input/output behavior of a DNN.	Frosst and Hinton (2017), Tan et al. (2018), Zhang et al. (2019a), Hou and Zhou (2020), Zhang et al. (2017, 2018), Harradon et al. (2018), Murdoch and Szlam (2017)

Table 2: Model distillation.

Figure 7: High level view of model distillation. The behavior of a trained deep learning model f used as training data for an explainable model g .

noise. From a practical perspective, this leads to heatmaps, indicating both positive and negative regions of interest for a specific class.

2.2 Model Distillation

Model distillation refers to a class of post-training explanation methods where the knowledge encoded within a trained DNN is *distilled* into a representation amenable for explanation by a user. This representation can take the form of more interpretable machine learning methods, e.g., decision trees. In this setting, as illustrated in Figure 7, an inherently transparent or white box machine learning model g is trained to mimic the input/output behavior of a trained opaque deep neural network f so that $g(\mathbf{x}) \approx f(\mathbf{x})$. Subsequent explanation of how g maps inputs to outputs may serve as a surrogate explanation of f 's mapping. Note that Hinton et al. (2015) outlines a method, with the same name, that implements a specific form of model distillation, namely distilling the knowledge learned by an ensemble of DNNs into a single DNN.

A distilled model in general learns to imitate the actions or qualities of an opaque DNN over the same data. It is a myth that the distilled form of a DNN necessarily underperforms compared to the original DNN (Liu et al., 2018a). Conceptually, this may be because a distilled model has access to information from the trained DNN, including the input

features it found to be most discriminatory and feature or output correlations relevant for classification. The distilled model can use this information directly during training, thus reducing the needed capacity of the distilled model. Since the distilled model still takes the original data as input, it may be further useful as a transparent view of how input features become related to the actions of the DNN. Interpreting the distilled model will not give insights into the internal representation of the data a DNN learns, or say anything about the DNN’s learning process, but can at least provide insight into the features, correlations, and relational rules that explain how the DNN operates. Put another way, the explanation of a distilled model can be seen as a hypothesis as to why a DNN has assigned some class label to an input.

Sometimes distillation methods appear similar to occlusion methods because distillation methods can also use occlusion in their algorithms. However, they differ in the following key aspect: While occlusion methods explicitly aim to make heatmaps, distillation methods aim to capture (local) model behavior by linking the information learned by occlusion with a more general representation. Moreover, compared to visualization methods, a much larger space of explanation forms become available to the practitioner that requires specialized knowledge about both the application domain as well as the interpretable model that will be distilled to. We organize model distillation techniques into two categories:

- **Local Approximation.** A local approximation method learns a simple model whose input/output behavior mimics that of a DNN for a small subset of the input data. This method is motivated by the idea that the model a DNN uses to discriminate within a local area of the data manifold is simpler than the discriminatory model over the entire surface. Given a sufficiently high local density of input data to approximate the local manifold with piecewise linear functions, the DNN’s behavior in this local area may be distilled into a set of explainable discriminators.
- **Model Translation.** Model translations train an alternative smaller model that mimics the input/output behavior of a DNN. They contrast local approximation methods in replicating the behavior of a DNN across an entire dataset rather than small subsets. The smaller models may be directly explainable, may be smaller and easier to deploy, or could be further analyzed to gain insights into the causes of the input/output behavior that the translated model replicates.

2.2.1 LOCAL APPROXIMATION

A local approximation method learns a distilled model that mimics DNN decisions on inputs within a small subset of the input examples. Local approximations are made for data subsets where feature values are very similar, so that a simple and explainable model can make decisions within a small area of the data manifold. While the inability to explain *every* decision of a DNN may seem unappealing, it is often the case that an analyst or practitioner is most interested in interpreting DNN actions under a particular subspace (for example, the space of gene data related to a particular cancer or the space of employee performance indicators associated with those fired for poor performance).

The idea of applying local approximations may have originated from Baehrens et al. (2010). These researchers present the notion of an “explainability vector”, defined by the

derivative of the conditional probability a datum is of a class given some evidence x_0 by a Bayes classifier. The direction and magnitude of the derivatives at various points x_0 along the data space define a vector field that characterizes flow away from a corresponding class. The work imitates an opaque classifier in a local area by learning a classifier that has the same form as a Bayes estimator for which the explanation vectors can be estimated.

Local Interpretable Model-Agnostic Explanations (LIME). Perhaps the most popular local approximation method is **LIME** developed by Ribeiro et al. (2016c). Figure 8 visually outlines the LIME process. From a *global*, black-box model f and an input of interest $\mathbf{x} \in \mathbb{R}^d$, LIME defines an interpretable model g from a class of *inherently* interpretable models $g \in G$ with different domain $\mathbb{R}^{d'}$ that approximates f well in the local area around \mathbf{x} . Examples of models in G may be decision trees or regression models whose weights explain the relevance of an input feature to a decision. Note that the domain of g is different from that of f . The model g operates over an interpretable representation \mathbf{x}' of the input data presented to the unexplainable model f , which could, for example, be a binary vector denoting the presence or absence of words in the text input, or a binary vector denoting if a certain pixel or color pattern exists in an image input. In Figure 8 examples of the interpretable representation \mathbf{x}' can be seen. In this case, \mathbf{x}' is a binary array indicating if a pixel belongs to a pattern or not by respectively assigning a 1 or a 0 to each pixel location. Noting that g could be a decision tree with very high depth or a regression model with many co-variate weights, an interpretable model that is overly complex may still not be useful or usable to a human. Thus, LIME also defines a measure of complexity $\Omega(g)$ on g . $\Omega(g)$ could measure the depth of a decision tree, the number of higher-order terms in a regression model, or it could be coded as if to check that a hard constraint is satisfied (e.g., $\Omega(g) = \infty$ if g is a tree and its depth exceeds some threshold). Let $\Pi_{\mathbf{x}}(\mathbf{z})$ be a similarity kernel between perturbed data point \mathbf{z} and a original data point $\mathbf{x} \in \mathbb{R}^d$ and a loss $\mathcal{L}(f, g, \Pi_{\mathbf{x}})$ defined to measure how poorly g approximates f on data in the area $\Pi_{\mathbf{x}}$ around the data point \mathbf{x} . To interpret $f(\mathbf{x})$, LIME identifies the model g satisfying:

$$\arg \min_g \{\mathcal{L}(f, g, \Pi_{\mathbf{x}}) + \Omega(g)\} \quad (18)$$

where $\Omega(g)$ serves as the complexity regularizer. In order for LIME to remain model agnostic, \mathcal{L} is approximated by uniform sampling over the non-empty space of $\mathbb{R}^{d'}$. For each sampled data point $\mathbf{z}' \in \mathbb{R}^{d'}$, LIME recovers the $\mathbf{x} \in \mathbb{R}^d$ corresponding to \mathbf{z}' , computes $f(\mathbf{z})$, and compares this to $g(\mathbf{z}')$ using \mathcal{L} . To make sure that the g minimizing Equation 18 fits well in the area local to the original reference point \mathbf{x} , the comparison of $f(\mathbf{z})$ to $g(\mathbf{z}')$ in \mathcal{L} is weighted by $\Pi_{\mathbf{x}}(\mathbf{z})$ so that samples farther from \mathbf{x} contribute less to the loss. The sampling process is repeated until the satisfactory dataset of locally perturbed samples $\mathcal{Z} = \{\mathbf{z}', f(\mathbf{z}), \Pi_{\mathbf{x}}(\mathbf{z})\}$ is obtained to train interpretable model g on.

The strength of LIME lies in the validation of the method with non-expert human practitioners. The original paper describes multiple experiments where non-expert users were asked to use the LIME explanations in different tasks. From an algorithmic perspective, LIME uses occlusion to find regions in the input that the model is sensitive to, similar to the perturbation-based methods discussed earlier. However, LIME aims to generalize the explanations in a local region around a reference image by learning an interpretable model.

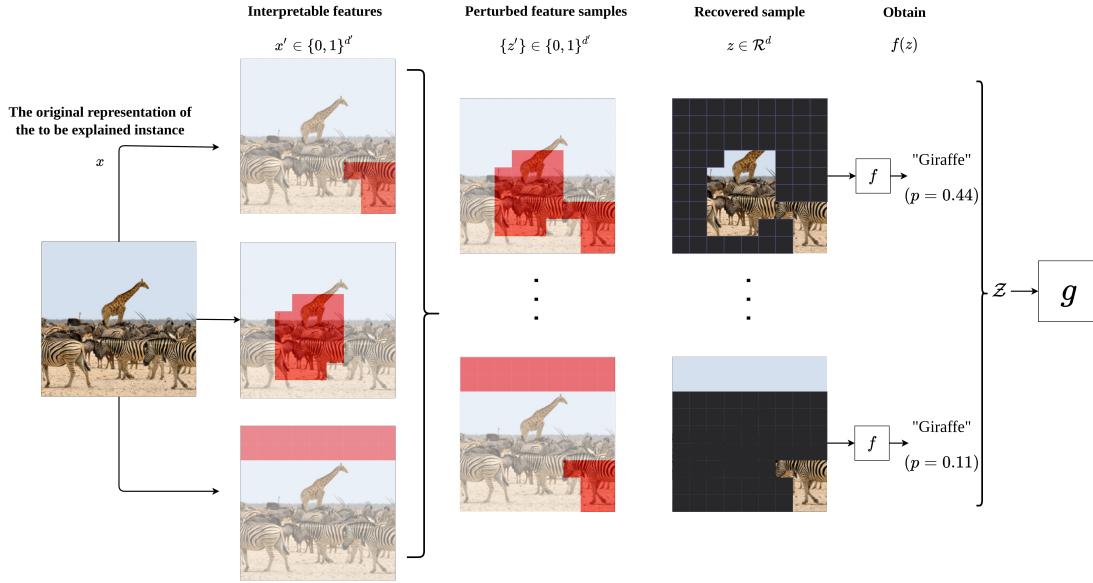


Figure 8: An image example explaining the process behind LIME. The interpretable representation \mathbf{x}' is a binary vector of super pixels, where each super pixel is a patch of contiguous pixels. \mathbf{z}' represents a random sampling of super pixels in \mathbf{x}' . \mathbf{z} is a new image in the same space as \mathbf{x} but where pixels not present in \mathbf{z}' are turned off. This process of sampling pixel subsets from the original input is repeated until the desired local perturbed dataset \mathcal{Z} around \mathbf{x} is collected. Finally \mathcal{Z} is used to train interpretable model g .

In practice, this means that the practitioner only needs one local LIME model for each set of similar inputs. In contrast to perturbation-based methods, where a new heatmap is computed for each image, once the practitioner has a local LIME model, assuming the input domain does not drastically change, the practitioner does not need to retrain a new local LIME model. This can be particularly useful when the data distribution has low variance.

We mention LIME in detail because other popular local approximation models (Ribeiro et al., 2016a, 2016b, 2018; Elenberg et al., 2017) follow LIME’s template and make their extensions or revisions. One drawback of LIME, which uses a linear combination of input features to provide local explanations, is that the precision and coverage of such explanations are not guaranteed. It is unclear if an explanation generated linearly and locally applies for a new data instance that might lie outside of the region where a linear combination of input features could represent. To address this issue, *anchor* methods (Ribeiro et al., 2016b, 2018) extend LIME to produce local explanations based on if-then rules, such that the explanations are locally anchored upon limited yet sufficiently stable input features for the given instance. Another drawback of LIME stems from the interpretable linear model’s training on a large set of randomly perturbed instances whose class label is assigned by the complex, opaque model. To reduce the time complexity, Elenberg et al. (2017) introduces STREAK, which

directly selects critical input components (for example, superpixels of images) by greedily solving a combinatorial maximization problem. Taking the image classification task as an example, an input image that is predicted as a class by the opaque model is first segmented into superpixels via image segmentation algorithm (Achanta et al., 2012). In every greedy step, a new superpixel is added to the set if containing it in the image will maximize the probability of the opaque model predicting the given class. The set of superpixels indicate the most important image regions that the opaque model used to make its decision.

Shapley Additive Explanations. More commonly referred to as SHAP (Lundberg & Lee, 2017), this method computes Shapley values (Shapley, 1953) for input feature sets. The Shapley value explanations are represented as the coefficients of a linear model. SHAP resembles perturbation-based methods because an incomplete (perturbed) input z is given to the model. The effects of the perturbation are measured, and a score is assigned based on the feature contribution amount. Only in this case, the contribution of *adding* a feature is measured and as opposed to measuring the *removal* of a feature as is the case in perturbation-based methods. The method uses a framework grounded in game theory that guarantees a unique solution of the additive feature attribution methods. According to the paper, various other explanation methods can be considered additive feature attribution methods. In SHAP, features are viewed as a member of a group. The method calculates how much each member contributes to the group. These contributions are called the Shapley values ϕ_i :

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S)] \quad (19)$$

where F is the set of all features, $S \subseteq F$, $f_{S \cup \{i\}}(x_{S \cup \{i\}})$ is a model trained with a subset of features that does *not* include feature x'_i and $f_S(x_S)$ is a model trained with a subset of features that does include feature x'_i . In practice this is implemented as:

$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!(M - |z'| - 1)!}{M!} [f_x(z') - f_x(z' \setminus i)] \quad (20)$$

where $z' \in \{0, 1\}^M$ is a simplified representation of the perturbed sample z indicating the presence of input features x'_i and M is the number of simplified input features. ϕ_i is calculated by sampling various combinations of features and measuring the change in prediction. In the end, a linear model g is fitted to the features and their effects:

$$g(z') = \phi_0 + \sum_{i=1}^M \phi_i z'_i \quad (21)$$

An example of this process is illustrated in Figure 9.

There exist different implementations of SHAP. KernelSHAP repeatedly samples features from the input, replacing a subset of the values by random values present in the data. This perturbed input is fed into the model, and the prediction is recorded. Each sampled feature set is assigned a weight using the SHAP kernel. After the sampling is done, a linear model is fitted. The Shapley values are extracted as the coefficients from

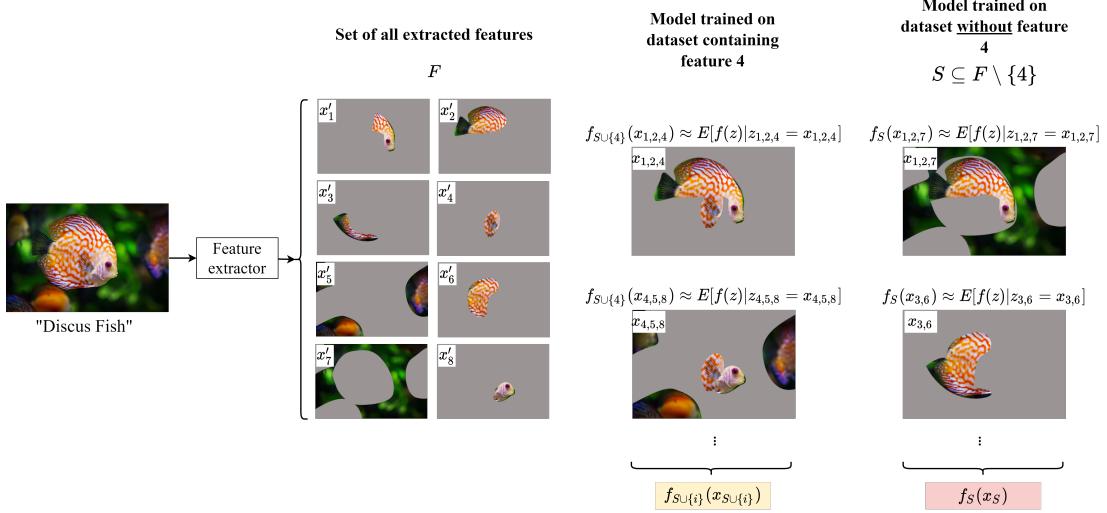


Figure 9: A visualization of the SHAP algorithm. In this example we have $M = 8$ simplified input features and the contribution of feature x'_4 is being measured.

the linear model. KernelSHAP samples from the marginal distribution and assumes that features are independent from each other. However, this is often not the case with real-world data such as naturalistic images. TreeSHAP was introduced as a faster alternative to KernelSHAP (Lundberg et al., 2018). Note that TreeSHAP is applicable only to tree-based ML models. The critical difference and why we mention TreeSHAP is that it samples from the conditional distribution instead of the marginal distribution, i.e., it does not assume that features are independent. Features that by themselves are not relevant for prediction can get a non-zero value assigned because they can now be correlated to other features relevant for model prediction. This is problematic because it violates the Shapley axiom stating that features that do not contribute to the prediction should have a value of zero. In practice, it means that the explanations produced by TreeSHAP are unreliable. Heskes et al. (2020) proposes a causal variation on SHAP called causal Shapley values. Causal Shapley values give a causal interpretation to Shapley values, allowing for the differentiation between direct and indirect feature contributions.

Even though SHAP is considered a local approximation method, we can run it multiple times to obtain global explanations. This is where SHAP becomes a compelling method since the global explanations are faithful to the local ones. Assuming sufficient Shapley values are calculated, SHAP becomes an explanation model in itself that can explain any instance. Strong examples are given by Molnar (2020).

As mentioned before, SHAP uses occlusion to find regions in the input that the model's output is sensitive to, similar to LIME. The other similarity that they share is that the explanation is an interpretable model. In the case of LIME, the choice of model is left to the practitioner, while with SHAP, the interpretable model is always a linear model as defined in Equation 21. SHAP's main strength comes from the way it defines an explanation as additive feature attribution grounded in a game-theoretical perspective. Viewed through this lens,

various explanation methods fit into this framework, and the relationships between methods become apparent. Like LIME, SHAP was also validated with real human practitioners, and the results show that SHAP explanations are better aligned with human intuition compared to several other methods, including LIME.

2.2.2 MODEL TRANSLATION

Compared to local approximation methods, model translation replicates the behavior of a DNN across an *entire* dataset rather than small subsets, through a smaller model that is easier for explanation. The smaller model could be easier to deploy (Hinton et al., 2015), faster to converge (Yim et al., 2017), or even be easily explainable, such as a decision tree (Frosst & Hinton, 2017; Bastani et al., 2017; Tan et al., 2018), Finite State Automaton (FSA) (Hou & Zhou, 2020), graphs (Zhang et al., 2017, 2018), or causal and rule-based classifier (Harradon et al., 2018; Murdoch & Szlam, 2017). We highlight the diversity of model types DNNs have been distilled into below.

Distillation to Decision Trees. Recent work has been inspired by the idea of tree-based methods for DNNs. Frosst and Hinton (2017) proposes “soft decision trees” which use stochastic gradient descent for training based on the predictions and learned filters of a given neural network. The performance of the soft decision trees is better than normal trees trained directly on the given dataset but is worse compared to the given pre-trained neural networks. Work proposed by Tan et al. (2018) generates global additive explanations for fully connected neural networks trained on tabular data through model distillation. Global additive explanations (Sobol, 2001; Hooker, 2004; Hoos & Leyton-Brown, 2014) have been leveraged to study complex models, including analyzing how model parameters influence model performance and decomposing black box models into lower-dimensional components. In this work, the global additive explanations are constructed by following previous work Hooker (2007) to decompose the black-box model into an additive model such as spline or bagged tree. They follow Craven and Shavlik (1996) to train the additive explainable model. Zhang et al. (2019a) trains a decision tree to depict the reasoning logic of a pre-trained DNN with respect to given model predictions. The authors first mine semantic patterns, such as objects, parts, and “decision modes” as fundamental blocks to build the decision tree. The tree is then trained to quantitatively explain which fundamental components are used for a prediction and the percentage of contribution respectively. The decision tree is organized in a hierarchical coarse-to-fine way, thus nodes close to the tree top correspond to common modes shared by multiple examples, while nodes at the bottom represent fine-grained modes with respect to specific examples.

From a practitioner’s perspective, distilling into decision trees has some advantages, arguably the most important one being that, compared to potentially subjective heatmaps, trees produce objective rules. Furthermore, formulating explanations as rules has the benefit that other algorithms can directly interpret the rules. The practitioner can then further automate the processing of the resulting explanations in their decision systems.

Distillation to Finite State Automata. Hou and Zhou (2020) introduces a new distillation of RNNs to explainable Finite State Automata (FSA). It is worth noting that this

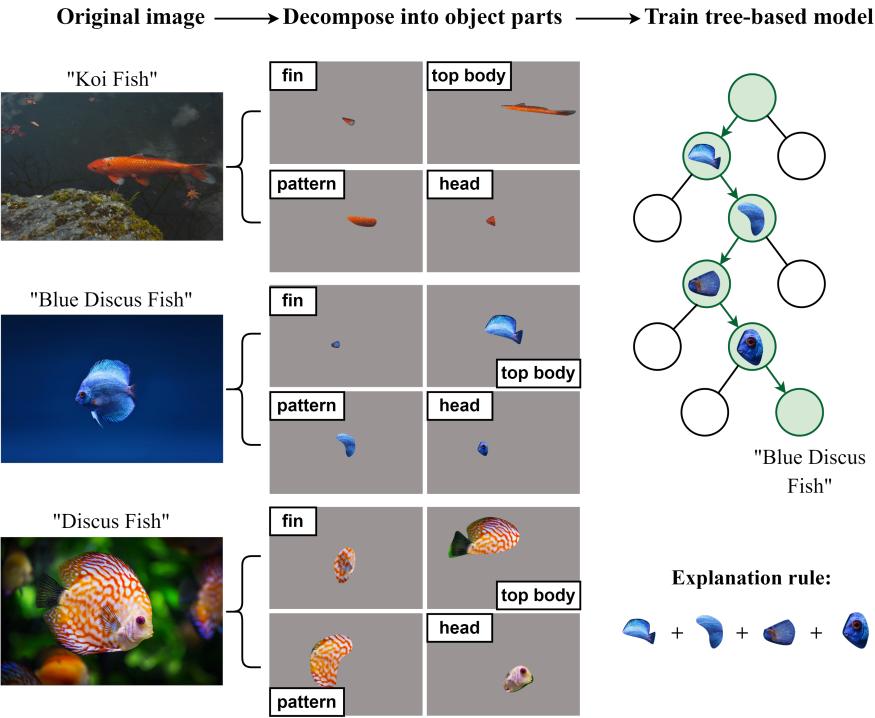


Figure 10: This example illustrated the general process of distilling a DNN into a decision tree. The specific details can differ, e.g., the practitioner can choose which feature extractor to use. Also the specific rules for training the tree-based model can differ. Furthermore, the type of tree can also differ. In this example the tree-based model is showing the explanation for "Blue Discuss Fish".

method is very specialized, only applicable to RNNs that perform binary classifications. An FSA consists of finite states and transitions between the states, and the transition from one state to another is a result of external input influence. FSA is formally defined as a 5-tuple $(\mathbb{E}, \mathbb{S}, s_0, \delta, \mathbb{F})$, where \mathbb{E} is a finite non-empty set of elements existing in input sequences, \mathbb{S} is a finite non-empty set of states, $s_0 \in \mathbb{S}$ is an initial state, $\delta : \mathbb{S} \times \mathbb{E} \rightarrow \mathbb{S}$ defines the state transmission function, and $\mathbb{F} \subseteq \mathbb{S}$ is the set of final states. The transition process of FSA is similar to RNNs in the sense that both methods accept items from some sequence one by one and transit between (hidden) states accordingly. The idea to distill an RNN to a FSA is based on the fact that the hidden states of an RNN tend to form clusters, which can be leveraged to build FSA. Two clustering methods, *k-means++* and *k-means-x* are adopted to cluster the hidden states of RNN towards constructing the explainable FSA model. The authors follow the structure learning technique and translate an RNN into an FSA, which is easier to interpret in two aspects, i) FSA can be simulated by humans; ii) the transitions between states in FSA have real physical meanings. Such a model translation helps to understand the inner mechanism of the given RNN model.

This is one of the few methods that try to understand the inner states of RNNs. The main benefit of distilling into an FSA is that an FSA can be represented graphically and in an objective manner. Similar to rules produced by a decision tree, this leads to objective explanations.

Distillation into Graphs. Both Zhang et al. (2017) and Zhang et al. (2018) build an object parts graph for a pre-trained CNN to provide model explanations. Similar to Zhang et al. (2019a), the authors first extract semantic patterns in the input and then gradually construct the graph for an explanation. Each node in the graph represents a part pattern, while each edge represents co-activation or spatial adjacent between part patterns. The explanatory graph explains the knowledge hierarchy inside the model, which can depict which nodes/part patterns are activated and the location of the parts in the corresponding feature maps.

The benefit of distilling into graphs is that graphs can capture and make transparent relational data. Information contained in the graph can be represented in several ways, e.g., as intuitive heatmaps that show which semantically interpretable part of the image was relevant to the prediction of the original model, or a graph that displays the connections between features in the input and how these connections relate to the model prediction.

Distillation into Causal and Rule-based Models. We also note work on distilling a DNN into symbolic rules and causal models. Harradon et al. (2018) constructs causal models based on concepts in a DNN. The semantics are defined over an arbitrary set of “concepts”, that could range from recognition of groups of neuron activations up to labeled semantic concepts. To construct the causal model, concepts of intermediate representations are extracted via an autoencoder. Based on the extracted concepts, a graphical Bayesian causal model is constructed to build association for the models’ inputs to concepts, and concepts to outputs. The causal model is leveraged to identify the input features of significant causal relevance with respect to a given classification result.

The practitioner can use this method to relate model predictions to the model’s learned concepts. So far, the other methods that do this as well were activation minimization and deconvolution. It is difficult to say whether one method is better than the other because the approaches vary significantly: activation minimization generates an image that maximally activates a target region in the network, sometimes producing strange images. Deconvolution uses translated convolutions to map the target activations back to the input, resulting in patterns representing concepts in the input. The difference between these two methods and Harradon et al. (2018) is that the latter visualizes concepts as heatmaps overlaid on the input, rather than generating a potentially uninterpretable image. However, this approach can sometimes lead to coarse explanations, especially if a concept is a relatively small part of the input. Another drawback is that the practitioner needs to provide semantic concept labels, which is not necessary with the two previously mentioned methods.

In another example, Murdoch and Szlam (2017) leverages a simple rule-based classifier to mimic the performance of an LSTM model. This study runs experiments on two natural language processing tasks, sentiment analysis, and question answering. The rule-based model is constructed via the following steps: i) decompose the outputs of an LSTM model, and generate important scores for each word; ii) based on the word level important score,

Intrinsic Methods	Comments	References
Attention Mechanisms	Leverage attention mechanisms to learn conditional distribution over given input units, composing a weighted contextual vector for downstream processing. The attention visualization reveals inherent explainability.	Bahdanau et al. (2015), Luong et al. (2015), Vaswani et al. (2017), Wang et al. (2016), Letarte et al. (2018), He et al. (2018), Devlin et al. (2019), Vinyals et al. (2015), Xu et al. (2015), Antol et al. (2015), Goyal et al. (2017), Teney et al. (2018), Mascharka et al. (2018), Anderson et al. (2018), Xie et al. (2019), Park et al. (2016)
Joint Training	Add additional explanation “task” to the original model task, and <i>jointly train</i> the explanation task along with the original task.	Zellers et al. (2019), Liu et al. (2019), Park et al. (2018), Kim et al. (2018b), Hendricks et al. (2016), Camburu et al. (2018), Hind et al. (2019), Melis and Jaakkola (2018), Iyer et al. (2018), Lei et al. (2016), Dong et al. (2017), Li et al. (2018a), Chen et al. (2019)

Table 3: Intrinsic methods.

important simple phrases are selected according to which jointly have high important scores; iii) The extracted phrase patterns are then used in the rule-based classifier, approximating the output of the LSTM model. Similar to using decision trees, rules have the advantage of being objectively interpretable and can be further processed as part of a decision system.

2.3 Intrinsic Methods

Ideally, we would like to have models that provide explanations for their decisions as part of the model output, or that the explanation can easily be derived from the model architecture. In other words, explanations should be *intrinsic* to the process of designing model architectures and during training. The ability for a network to intrinsically express an explanation may be more desirable compared to post-hoc methods that seek explanations of models that were never designed to be explainable in the first place. This is because an intrinsic model has the capacity to learn not only accurate outputs per input but also outputs expressing an explanation of the network’s action that is optimal with respect to some notion of explanatory fidelity. Ras et al. (2018) previously defined a category related to this approach as *intrinsic methods* and identified various methods that offer explainable extensions of the model architecture or the training scheme. In this section, we extend the notion of intrinsic explainability with models that actually provide an explanation for their decision even as they are being trained. Figure 11 shows the difference between the process of deriving explanations post-hoc compared to intrinsic explanations. The main difference is that with intrinsic explanations, the explanations are part of the model or part of the model output, giving the practitioner additional information during the model training phase that is not available during the post-hoc derivation of explanation.

Like distillation methods, the practitioner needs to have explicit knowledge about the field of application and the models required to learn explanations intrinsically. Intrinsic methods are arguably the most difficult methods to apply compared to visualization and distillation methods because the practitioner needs additional specialized knowledge and because the training process will involve multiple models that can make optimization more challenging. Additionally, more computing power is likely required to train a larger pipeline of joint models.

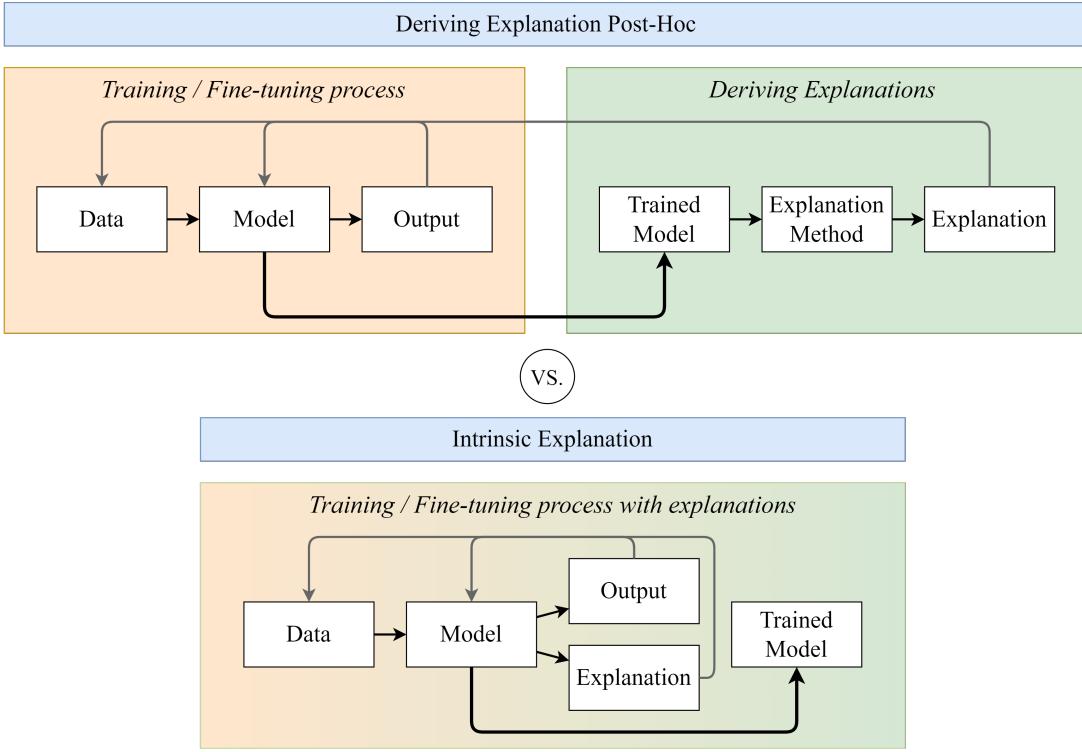


Figure 11: A process comparison between deriving an explanation post-hoc vs. a model where the explanations are intrinsic. The goal is to produce a trained, explainable model. When extracting explanations post-hoc, the process consists of separate training and explanation stages. The explanations can potentially guide the original model for performance improvement (grey arrows going backward). For intrinsic explanation, generating explanations is integrated into the training process. It is worth noting that using intrinsic explanations could be more time-consuming since the model needs to learn a more complex problem.

We observe methods in the literature on intrinsically explainable DNNs to follow two trends: (i) they introduce *attention mechanisms* to a DNN, and the attention visualization reveals inherent explainability; (ii) they add additional explanation “task” to the original model task, and *jointly train* the explanation task along with the original task. We explain the trends and highlight the representative methods below.

2.3.1 ATTENTION MECHANISMS

DNNs can be endowed with attention mechanisms that simultaneously preserve or even improve their performance and have explainable outputs expressing their operations. An attention mechanism (Vaswani et al., 2017; Devlin et al., 2019; Teney et al., 2018; Xie et al., 2019) learns a conditional distribution over given input units, composing a weighted contextual vector for downstream processing. The attention weights can be generated in

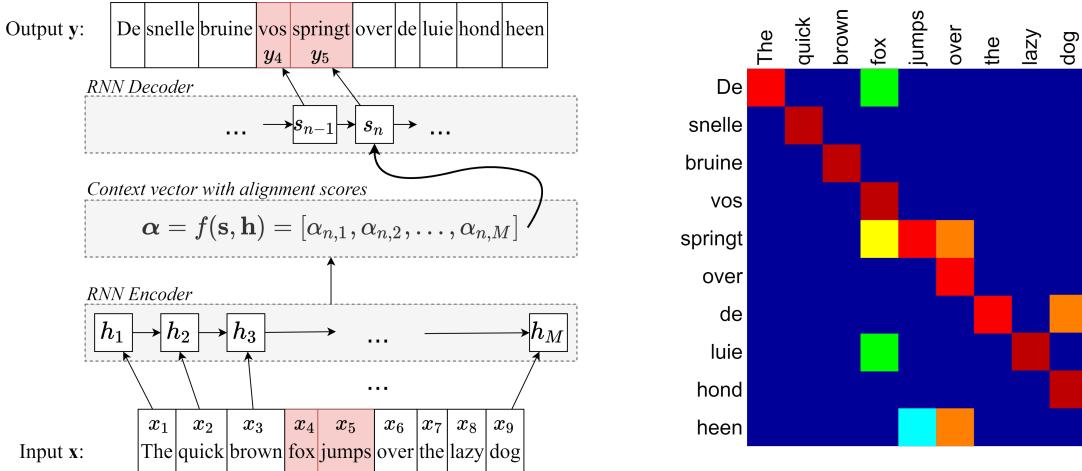


Figure 12: **Left:** High-level overview of using attention as explanation. The model takes an English sentence as input and outputs a Dutch translation. During the forward pass, the alignment scores α (attention weights) are calculated as part of the training process and can immediately be visualized as a heatmap. The α maps the correlation between the different parts of the input (hidden states \mathbf{h}) and output (hidden states \mathbf{s}). Alignment score function f determines how α is computed. For a detailed explanation on how attention works, see Bahdanau et al. (2015). **Right:** The attention matrix where each value is the alignment score $\alpha_{n,m}$ between encoder hidden state h_m and decoder hidden state s_n .

multiple ways, such as by calculating cosine similarity (Graves et al., 2014), adding additive model structure, such as several fully connected layers, to explicitly generate attention weights (Bahdanau et al., 2015), leveraging the matrix dot-product (Luong et al., 2015) or scaled dot-product (Vaswani et al., 2017), and so on. Attention mechanisms have shown to improve DNN performance for particular types of tasks, including tasks on ordered inputs as seen in natural language processing (Vaswani et al., 2017; Devlin et al., 2019) and multi-modal fusion such as visual question answering (Anderson et al., 2018).

Single-Modal Weighting. The output of attention mechanisms during a forward pass can inform a practitioner about how different input features are weighted at different phases of model inference. In pure text processing tasks such as language translation (Bahdanau et al., 2015; Luong et al., 2015; Vaswani et al., 2017) or sentiment analysis (Wang et al., 2016; Letarte et al., 2018; He et al., 2018), attention mechanisms allow the downstream modules, a decoder for language translation or fully connected layers for classification tasks, to concentrate on different words in the input sentence by assigning learned weights to them (Vaswani et al., 2017; Wang et al., 2016). To provide straightforward explanations, the attention weights can be visualized as heatmaps, depicting the magnitude and the sign (positive or negative) of each weight value, showing how input elements weighted combined to influence the model latter processing and the final decisions.

Figure 12 gives a visual example of how attention can be used for explanation for single-modal weighting. The left-hand side of the figure depicts a basic RNN encoder-decoder architecture where the dense representation of text input is associated with the attention weights. These weights can be plotted as a matrix, and each input’s importance as it relates to the output can directly be interpreted. This type of explanation can help the practitioner monitor model predictions during training and give insight into whether the model utilizes undesirable correlations in the dataset. The practitioner needs to keep in mind that it will become cumbersome to monitor individual model predictions as the input size increases.

Multi-Modal Interaction. In multi-modal interaction tasks, such as image captioning (Vinyals et al., 2015; Xu et al., 2015), visual question answering (Antol et al., 2015; Goyal et al., 2017; Johnson et al., 2017; Teney et al., 2018) or visual entailment (Xie et al., 2019), attention mechanisms play an important role in feature alignment and fusion across different feature spaces (for instance, between text and images). For example, Park *et al.* propose the Pointing and Justification model that uses multiple attention mechanisms to explain the answer of a VQA task with natural language explanations and image region alignments (Park et al., 2016). Xie *et al.* use attention mechanisms to recover semantically meaningful areas of an image that correspond to the reason a statement is, is not, or could be entailed by the image’s conveyance (Xie et al., 2019). Mascharka et al. (2018) aims to close the gap between performance and explainability in visual reasoning by introducing a neural module network that explicitly models an attention mechanism in image space. By passing attention masks between modules it becomes explainable by being able to directly visualize the masks. This shows how the attention of the model shifts as it considers the different components of the input.

Multi-modal interaction methods go one step beyond single-modal weighting by combining multiple attention mechanisms with supplementary tasks that increase (i) the model’s interpretability and (ii) give the practitioner additional opportunities for creating explanations that cater to the area of application. However, compared to single-modal weighting, multi-modal interaction can be more difficult to apply due to the higher complexity accompanying the increased combination of attention components and multiple tasks.

2.3.2 JOINT TRAINING

This type of intrinsic method introduces an additional “task” besides the original model task, and jointly trains the additional task together with the original one. Here we generalize the meaning of a “task” by including preprocessing or other steps involved in the model optimization process. The additional task is designed to provide model explanations directly or indirectly. Such additional task can be in the form of i) text explanation, which is a task that directly provides explanations in natural language format; ii) explanation association, which is a step that associates input elements or latent features with human-understandable concepts or objects, or even directly with model explanations; iii) model prototype which learns a prototype that has clear semantic meanings as a preprocessing step. Explanations are generated based on the comparison between the model behavior and the prototype.

In Figure 13 we see a high-level overview of the joint training with, in this case, an example of an image captioning task. The objective $\arg \min_{\theta} \frac{1}{N} \sum_{n=1}^N \alpha \mathcal{L}(y_n, y') + \beta \mathcal{L}(e_n, e')$

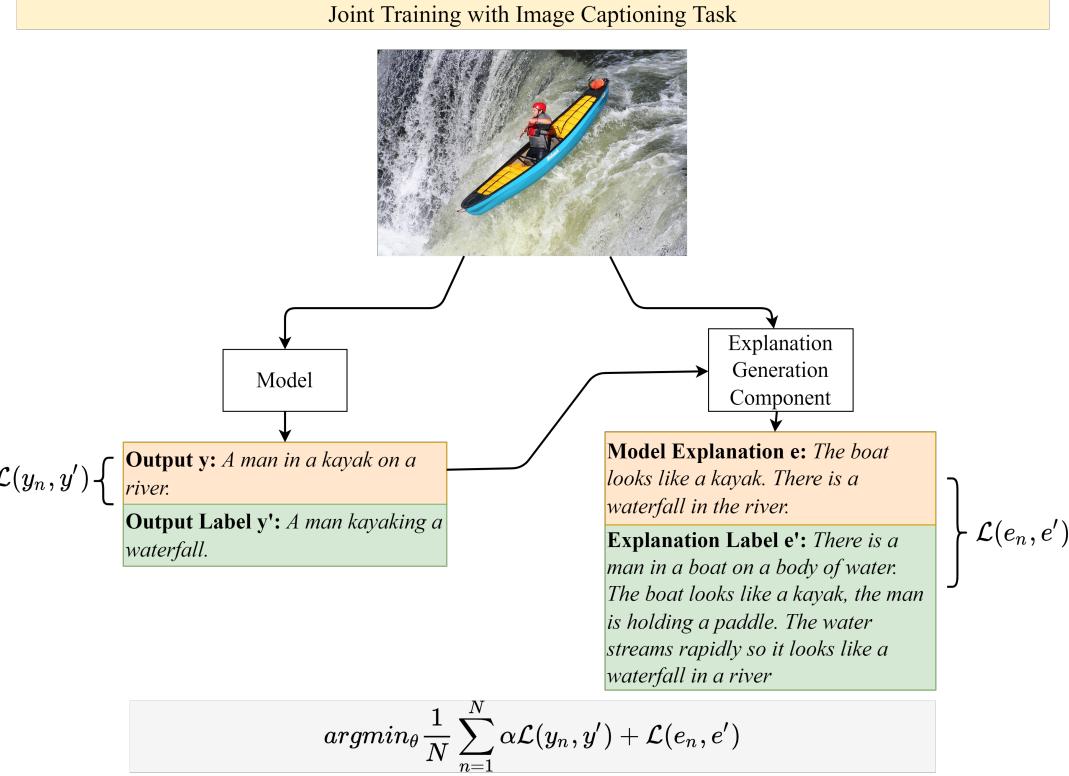


Figure 13: High level overview of how joint training works. The algorithm tries to minimize the average combined weighted loss over the output prediction and the explanation generation, where α denotes the weight.

is a very general form of the function that has to be minimized and is composed of at least two losses: the prediction loss and the explanation component loss. By weighting each loss, a balance can be found between having a model that gives good predictions and a model that gives good explanations.

Text Explanation. A group of recent work (Zellers et al., 2019; Liu et al., 2019; Park et al., 2018; Kim et al., 2018b; Hendricks et al., 2016; Camburu et al., 2018; Hind et al., 2019) achieve the explainable goal via augmenting the original DNN architecture with an explanation generation component and conducting joint training to provide natural language explanations along with the model decisions, similar to what is illustrated in Figure 13. Such explainable methods are straightforward and the explanations that they produce are layman-friendly since they are presented in natural language, instead of figures or statistical data. The explanation could be either generated word by word similar to a sequence generation task (Hendricks et al., 2016; Park et al., 2018; Camburu et al., 2018; Kim et al., 2018b; Liu et al., 2019), or be predicted from multiple candidate choices (Zellers et al., 2019).

The primary advantage of joint training text explanations is that the practitioner can tailor the explanations to the users' needs while using state-of-the-art models. This way, the practitioner can get the best of both worlds. On the other hand, obtaining the appropriate (labeled) dataset for the explanation generation component is difficult and time-consuming. Furthermore, the practitioner must overcome the additional difficulties that joint training presents when training multiple models jointly instead of training a single model. Finally, it is known that the generated explanations exhibit some inconsistencies (Oana-Maria et al., 2020) which undermines the trust in the explanations provided by the model.

Hendricks et al. (2016) is an early work that provides text justifications along with its image classification results. The approach combines image captioning, sampling, and deep reinforcement learning to generate textual explanations. The class information is incorporated into the text explanations, which makes this method distinct from normal image captioning models that only consider visual information, via i) include class as an additional input for text generation and ii) adopt a reinforcement learning based loss that encourages generated sentences to include class discriminative information.

Liu et al. (2019) proposes a Generative Explanation Framework (GEF) for text classifications. The framework is designed to generate fine-grained explanations such as text justifications. During training, both the class labels and fine-grained explanations are provided for supervision. The loss of GEF contains two parts, classification loss and explanation generation loss. To make the generated explanations class-specific, an “explanation factor” is designed in the model structure to associate explanations with classifications. The “explanation factor” is intuitively based on directly taking the explanations as input for classification and adding constraints on the classification softmax outputs.

Unlike previous methods which *generate* text explanations, Zellers et al. (2019) provides explanations in a *multichoice* fashion. They propose a visual reasoning task named Visual Commonsense Reasoning (VCR), which is to answer text questions based on given visual information (image), and provide reasons (explanations) accordingly. Both the answers and reasons are provided in a multichoice format. Due to the multichoice nature, reasonable explanations should be provided during testing, in contrast to other works which could generate explanations along with model decisions. VCR is thus more suitable for prototype model debugging to audit the model reasoning process.

Explanation Association. This type of joint training method associates input elements and latent features with human-understandable concepts or objects (Melis & Jaakkola, 2018; Iyer et al., 2018; Lei et al., 2016; Dong et al., 2017). Such methods usually achieve explanations by adding a regularization term (Melis & Jaakkola, 2018; Lei et al., 2016; Dong et al., 2017) and/or revising a model’s architecture (Melis & Jaakkola, 2018; Iyer et al., 2018; Lei et al., 2016). The explanations are provided in the form of i) associating input features or latent activations with semantic concepts (Melis & Jaakkola, 2018; Dong et al., 2017); ii) associating model prediction with a set of input elements (Lei et al., 2016); iii) associating explanations with object saliency maps in a computer vision task (Iyer et al., 2018). Regardless of the format of explanations and the technical details, methods belonging to this type commonly share the characteristics of learning to associate hard-to-interpret elements with human-understandable atoms intrinsically.

Melis and Jaakkola (2018) proposes an intrinsic method which associates input features with semantically meaningful concepts and regards the coefficient as the importance of such concepts during inference. A regularization based general framework for creating self-explaining neural networks (SENNs) is introduced. Given raw input, the network jointly learns to generate the class prediction and to generate explanations in terms of an input feature-to-concept mapping. The framework is based on the notion that linear regression models are explainable and generalizes the respective model definition to encompass complex classification functions, such as a DNN. A SENN consists of three components: i) A concept encoder that transforms the raw input into a set of explainable concepts. This encoder can be understood as a function that transforms low-level input into high-level meaningful structure, which predictions and explanations can be built upon. ii) An input-dependent parametrizer, a procedure to get the coefficient of explainable concepts and learn the relevance of explainable concepts for class predictions. The values of relevance scores quantify the positive or negative contribution of the concept to the prediction. iii) An aggregation function (e.g., a sum) that combines the output of the concept encoder and the parametrizer to produce a class prediction.

Iyer et al. (2018) introduces Object-sensitive Deep Reinforcement Learning (O-DRL), which is an explanation framework for reinforcement learning tasks that takes videos as input. O-DRL adds a pre-processing step (template matching) to recognize and locate specific objects in the input frame. For each detected object, an extra channel is added to the input frame’s RGB channels. Each object channel is a binary map with the same height and width as the original input frame, 1’s encoding for the detected object’s location. The binary maps are later used to generate object saliency maps (as opposed to pixel saliency maps) that indicate the relevance of the object to action generation. It is argued that object saliency maps are more meaningful and explainable than pixel saliency maps since the objects encapsulate a higher-level visual concept.

Lei et al. (2016) integrates explainability in their neural networks for sentiment analysis by learning rationale extraction during the training phase in an unsupervised manner. Rationale extraction allows the network to learn to identify a small subset of words that all lead to the same class prediction as the entire text. They achieve this by adding mechanisms that use a combination of a generator and an encoder. The generator learns which text fragments could be candidate rationales, and the encoder uses these candidates for prediction. Both the generator and the encoder are jointly trained during the optimization phase. The model explanation is provided by associating the model prediction with a set of critical input words.

Dong et al. (2017) focuses on providing intrinsic explanations for models on video captioning tasks. An interpretive loss function is defined to increase the visual fidelity of the learned features. This method is based on the nature of the dataset, which contains rich text descriptions along with each video. To produce an explanation, semantically meaningful concepts are first pre-extracted from human descriptions via Latent Dirichlet Allocation, which covers a variety of visual concepts such as objects, actions, and relationships. Based on the pre-extracted semantic topic, an interpretive loss is added to the original video captioning DNN model, for jointly training to generate video captions along with forcing the hidden neurons to be associated with semantic concepts.

From a practitioner’s perspective, explanation association can be powerful because se-

mantically meaningful concepts are directly interpretable by humans. Explanations based on these concepts can be represented in various forms, e.g., a relational graph or a heatmap, unlike text explanations which are limited to text only. Similar to text explanations, explanation association can also require specialized (labeled) datasets, which are difficult to obtain. Unlike text explanations, which mostly use an external component to generate the explanations, explanation associations often use internal model representations to associate high-level concepts. The practitioner may need to modify existing models to gain access to these internal representations, or in some cases, the internal representations might not be accessible to the practitioner. Furthermore, various methods in this category require a pipeline of several models where each model can become a bottleneck in the joint training process.

Model Prototype. This type of intrinsic method is specifically for classification tasks, and is derived from a classical form of case-based reasoning (Kolodner, 1992) called prototype classification (Marchette & Socolinsky, 2003; Bien & Tibshirani, 2011; Kim et al., 2014). A prototype classifier generates classifications based on the similarity between the input and each prototype observation in the dataset. In prototype classification applications, the word “prototype” is not limited to an observation in the dataset but can be generalized to a combination of several observations or a latent representation learned in the feature space. The model architecture is designed to enable joint training of the prototypes and the original task to provide intrinsic explanations. The model explainability is achieved by tracing the reasoning path for the given prediction back to each prototype learned by the model.

Li et al. (2018a) proposes an explainable prototype-based image classifier that can trace the model classification path to enable reasoning transparency. The model contains two major components: an autoencoder and a prototype classifier. The autoencoder transforms raw input into a latent feature space, and the prototype classifier uses the latent features for classification. The prototype classifier, on the other hand, generates a classification via i) first calculating the distances in the latent space between a given input image and each prototype, ii) then passing through a fully-connected layer to compute the weighted sum of the distances, and iii) finally normalizing the weighted sums by the softmax layer to generate the classification result. Because the network learns *prototypes* during the training phase, each prediction always has an explanation that is faithful to what the network computes. Each prototype can be visualized by the decoder, and the reasoning path of the prototype classifier can be partially traced given the fully-connected layer weights and the comparison between input and each visualized prototype, providing intrinsic model explanations.

Chen et al. (2019) introduces an explainable DNN architecture called Prototypical Part Network (ProtoPNet) for image classification tasks. Similar to Li et al. (2018a), ProtoPNet also contains two components; a regular convolutional neural network and a prototype classifier. The regular convolutional neural network projects the raw image into hidden feature space, where prototypes are learned. The prototype classifier generates model predictions based on the weighted sum of each similarity score between an image patch and a learned prototype. Unlike Li et al. (2018a) where learned prototypes correspond to the entire image, the prototypes in (Chen et al., 2019) are more fine-grained and are latent representations of parts/patches of the image. To provide a model explanation, the latent representation of

each prototype is associated with an image patch in the training set, shedding light on the reasoning clue of ProtoPNet.

Compared to the other methods in this category, prototypes adopt a different approach towards creating interpretable model architecture. Other methods tend to use existing model architectures and make adaptations that either grant the practitioner access to either internal model components or add models to the existing model pipeline. In the case of prototypes, the practitioner often creates a novel architecture with traceable paths of reasoning. In some sense, model prototypes avoid end-to-end architectures where a single DNN learns the entire task. Instead, the DNN is constructed having explicitly interpretable components baked in as part of its design.

2.4 Summary

Methods discussed in this field guide are categorized by distinct philosophies on eliciting and expressing an explanation from a DNN. This organization is ideal to understand the “classes” of methods that are being investigated in research and gradually implemented in practice. This does not, however, resolve an obvious question from a machine learning practitioner: *What is the “right” type of explanatory method I should use when building a model to solve my specific kind of problem?* It is difficult to match the methods with a particular situation because the type of explanation method suitable to that particular situation is often dependent on many variables including the type of DNN architecture, data, problem, and desired form of explanation.

We propose Table 4 and Table 5 as a starting point in answering this question. All of the papers in Figure 3 are organized in Table 4 and Table 5. Each table is titled with a category of explanation method. Both tables are organized into five columns. The first column indicates the subcategory of the explanation method, and the second column displays the reference to the explanation paper. The following three columns contain summarized information taken directly from the explanation paper. The third and fourth columns contain one or more icons representing the type of data used and the type of problem(s) presented in the paper respectively. The meaning of the icons can be found at the bottom of the table. The final column displays information about the specific DNN model on which the explanation method has been used in the paper.

The practitioner can make use of Table 4 and Table 5 by considering what type of data, problem, and DNN architecture they are using in their specific situation. Then the practitioner can find an appropriate explanation method by matching the type of data, problem, and DNN architecture with the ones in the tables. For example, if a practitioner is using an image dataset to train a CNN on a classification problem, the practitioner can consider explanation methods for which the  icon and the  icon and “CNN” are present in the respective rows. Note that in the “DNN Type” column we use “no specific requirements” to indicate that the DNN used in the respective paper does not need to meet any other specific requirements other than being a DNN. We use “model agnostic” to indicate that the type of model does not matter, i.e., the model does not have to be a DNN.

a) Visualization Methods

	Explanation Paper	Data Type	Problem Type	DNN Type
Back-Propagation	Erhan et al. (2009)	☒	○	classifier has to be differentiable
	Zeiler et al. (2011)	☒	○	CNN with max-pooling + relu
	Zeiler and Fergus (2014)	☒	○	CNN with max-pooling + relu
	Selvaraju et al. (2017)	☒	○ ⚪ 🐾 🏷	CNN
	Zhou et al. (2016)	☒ 📄	○ ⚪ ⚪ 🐾 🏷	CNN with global average pooling + softmax output
	Bach et al. (2015)	☒ 📄	○	multilayer network
	Lapuschkin et al. (2016)	☒	○	CNN
	Arras et al. (2016)	☒	○	CNN
	Arras et al. (2017)	☒	○	CNN
	Ding et al. (2017)	📄	➡	attention-based encoder decoder
	Montavon et al. (2017)	agnostic	○	no specific requirements
	Shrikumar et al. (2017)	☒ 📄	○	CNN
	Sundararajan et al. (2017)	☒ 📄 🌱	○ ➡	no specific requirements
	Sundararajan et al. (2016)	☒ 📄 🌱	○	no specific requirements
Perturbation	Zeiler and Fergus (2014)	☒	○	CNN
	Li et al. (2016)	☒ 📄	○ 🛍	no specific requirements
	Fong and Vedaldi (2017)	☒	○	model agnostic
	Zintgraf et al. (2017)	☒	○	CNN
	Robnik-Šikonja and Kononenko (2008)	☰	○	models has to output probabilities
	Dabkowski and Gal (2017)	☒	○	classifier has to be differentiable

b) Model Distillation

	Loc. Appr.	Data Types	Problem Types	DNN Type
Loc. Appr.	Ribeiro et al. (2016c)	☒ 📄 ☰	○	model agnostic
	Ribeiro et al. (2016b)	☒ 📄 ☰	○	model agnostic
	Ribeiro et al. (2018)	☒ 📄 ☰ ☰	○ 🛢️ ✎	model agnostic
	Elenberg et al. (2017)	☒ ☰	○	model agnostic
	Baehrens et al. (2010)	☒ 📄 ☰	○	model agnostic
	Lundberg and Lee (2017)	☒ 📄 ☰	○	model agnostic
	Heskes et al. (2020)	☒ 📄 ☰	○	model agnostic
Model Translation	Hou and Zhou (2020)	📄	○	RNN
	Murdoch and Szlam (2017)	📄	○ 🗣️	LSTM
	Harradon et al. (2018)	☒ ☰	○	CNN
	Frosst and Hinton (2017)	☒ ☰	○	CNN
	Zhang et al. (2019a)	☒ ☰	○	CNN
	Tan et al. (2018)	☒ ☰	○ ⚪	no specific requirements
	Zhang et al. (2017)	☒ ☰	○	CNN
	Zhang et al. (2018)	☒ ☰	○	CNN, GANs

Data Types	Problem Types
☒ image	○ classification
📄 text	📍 localization
☰ molecular graph	⚡ visual question answering
☒ DNA sequence	ⓧ regression
☒ embedding	ⓧ language translation
☒ categorical data	ⓧ sequence tagging
☒ tabular data	ⓧ structured prediction
	ⓧ text generation
	ⓧ question answering
	ⓧ captioning

Table 4: Lookup table for the (a) visualization and (b) model distillation methods.

Intrinsic Methods				
	Explanation Paper	Data Type	Problem Type	DNN Type
Attention Mechanisms	Vaswani et al. (2017)	█	↻	transformer
	Devlin et al. (2019)	█	A	transformer
	Bahdanau et al. (2015)	█	⊕	RNN encoder-decoder
	Luong et al. (2015)	█	⊕	stacking LSTM
	Wang et al. (2016)	█	♥	attention-based LSTM
	Letarte et al. (2018)	█	♥ C	self-attention network
	He et al. (2018)	█	♥	attention-based LSTM
	Teney et al. (2018)	█ █	🐘	CNN + GRU combination
	Mascharka et al. (2018)	█	🐘	various specialized modules
	Xie et al. (2019)	█	⌚ ⊕	various specialized modules
	Park et al. (2016)	█ █	⌚ ⊕ 🐘	various specialized modules
	Vinyals et al. (2015)	█	⌚ ⊕ 🐘	CNN + LSTM combination
	Xu et al. (2015)	█	⌚ ⊕ 🐘	CNN + RNN combination
	Antol et al. (2015)	█ █	🐘	CNN + MLP, CNN + LSTM combinations
Joint Training	Goyal et al. (2017)	█ █	🐘	CNN + LSTM combination
	Anderson et al. (2018)	█ █	🐘 C	region proposal network + resnet combo, LSTM
	Camburu et al. (2018)	█	A	LSTM
	Hind et al. (2019)	█	C	model agnostic
	Hendricks et al. (2016)	█	C	CNN
	Zellers et al. (2019)	█ █	⌚	recognition to cognition network
	Liu et al. (2019)	█	C	encoder-predictor
	Park et al. (2018)	█ █	C 🐘	pointing and justification model
	Kim et al. (2018b)	█	✗	CNN
	Lei et al. (2016)	█	♥	encoder-generator
	Melis and Jaakkola (2018)	█ ⚡田	C	self-explaining neural network
	Iyer et al. (2018)	█	🤖	deep q-network
	Dong et al. (2017)	█	cc	attentive encoder-decoder
	Li et al. (2018a)	█	C	autoencoder + prototype layer combination
	Chen et al. (2019)	█	C	prototypical part network

Data Types	Problem Types
█ image	⌚ classification
█ text	🐘 visual question answering
█ █ embedding	⊕ language translation
█ █ categorical data	⌚ captioning
█ ⚡ tabular data	♥ sentiment analysis
█ video	⌚ visual entailment
	⌚ visual commonsense reasoning
	⌚ language understanding
	⌚ reinforcement learning
	⌚ control planning

Table 5: Lookup table for the intrinsic methods.

3. Evaluating Explanations

There is a growing body of research on the objective comparison of explanation methods and their quality. It is important to be able to evaluate the factual quality of generated explanations. Evidence suggests that when humans and AI collaborate, humans often make better decisions when the AI provides a correct explanation (Ray et al., 2019). When the explanation is incorrect it can lead to bad outcomes (Jacovi & Goldberg, 2020). In addition, practitioners need to know whether they can trust the explanation that the methods return. It is well known that explanation methods are subject to misinterpretation, especially visual explanation methods (Kindermans et al., 2019; Nie et al., 2018; Adebayo et al., 2018; Alvarez-Melis & Jaakkola, 2018). There are concerns regarding the factual correctness of explanation methods such as deconvolution, guided backpropagation and LRP (Kindermans et al., 2018). One main challenge for explanation evaluations is the lack of ground truth for most cases. In addition, a favorable evaluation metric may vary by the specific evaluation goal and the user group an explanation is designed for.

3.1 What Makes a Good Explanation?

The answer to this question depends on the user, the context of use, the type of model and data, and the desired explanation form. The literature has come up with various desiderata (Ras et al., 2018; Robnik-Šikonja & Bohanec, 2018; Carvalho et al., 2019; Jacovi & Goldberg, 2020), with the traits *fidelity*, *consistency*, *stability* and *comprehensibility* most commonly scrutinized and discussed.

3.2 Methods for Evaluating Explanation Methods and their Explanations

There are practically two main approaches for evaluating explanations. The first is to devise an objective metric or benchmark to evaluate the explanations without human intervention (Samek et al., 2016; Hooker et al., 2019; Vu et al., 2019; Adebayo et al., 2018; Alvarez-Melis & Jaakkola, 2018). This approach has the benefit of being able to compare numerous explanation methods with each other. Using objective benchmarks we can investigate to what extent desiderata such as fidelity, consistency and stability are being satisfied. Given that visualization methods that produce heatmaps are a popular and intuitive type of explanation method, it has gained most of the attention in the subfield of evaluation explanation. Specifically, evaluating heatmaps generated for image classification networks is the focus of various evaluation work. There also exists a small body of work in evaluating explanations in NLP (DeYoung et al., 2020). The second approach is to let a human evaluate the explanations (Prasad et al., 2020; Hase & Bansal, 2020; Jesus et al., 2021). By using humans to evaluate the explanations, we can investigate to what degree the following desiderata are satisfied: clarity, parsimony, comprehensibility and importance.

3.2.1 EVALUATING HEATMAPS

Even though the following evaluations focus on heatmaps derived from image classifiers, they can also be applied to heatmaps of text in NLP explanations, e.g., LIME or SHAP. The process pipeline for some of the methods described below are illustrated in Figure 14. This figure makes it easy to compare the different ways of evaluating heatmaps and demonstrated that

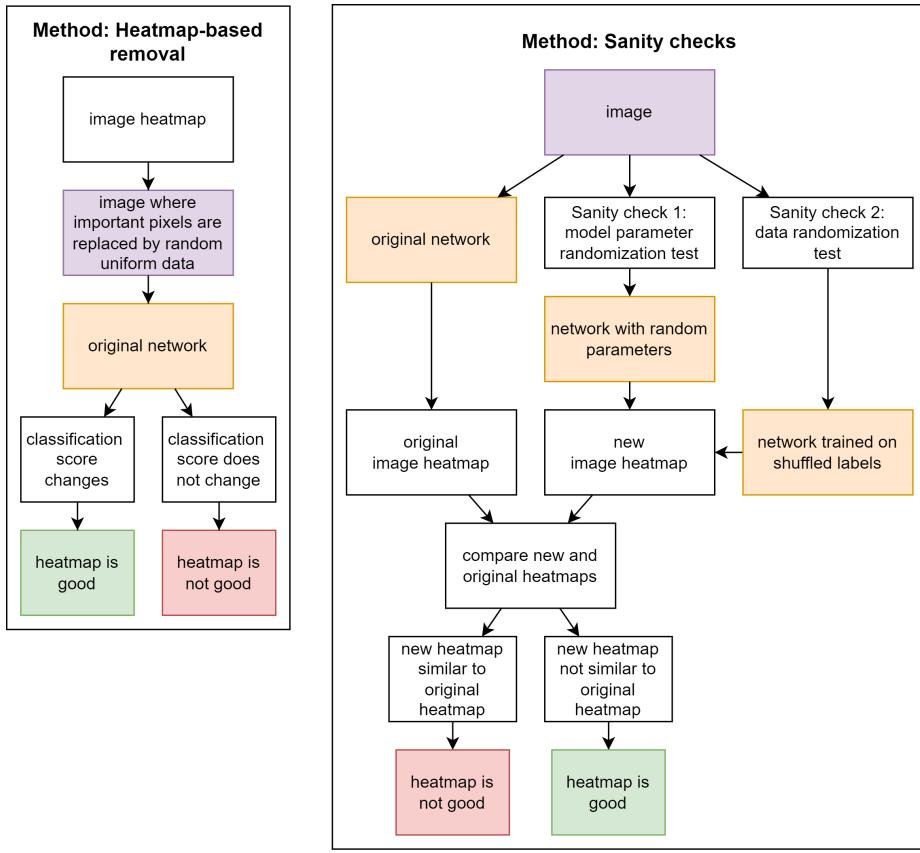


Figure 14: The diagram on the left-hand illustrates a straightforward process where areas indicated as important by the heatmap are replaced by random pixels. On the right-hand side the process is more elaborate, evaluating two specific sensitivities of the heatmap by performing two different checks. Each check evaluates a specific property of the heatmap.

the process of evaluating heatmaps can occur in numerous ways depending on the evaluation metric. Samek et al. (2016) and variations (Kindermans et al., 2018; Petsiuk et al., 2018) introduce a perturbation-based method for evaluating the quality of heatmaps. Using the method, they compare the quality of heatmaps generated by sensitivity analysis (Simonyan et al., 2013), deconvolution (Zeiler & Fergus, 2014) and LRP (Bach et al., 2015) by replacing the regions of the image that correspond to the location of the heatmap with randomly uniform data and checking how much the classification score changes. According to their metric, the more the classification score changes, the better the heatmap corresponds to class-discriminating features. Their results show that the heatmaps produced by LRP correspond better to the class features than heatmaps produced by sensitivity analysis and deconvolution.

However, Hooker et al. (2019) argues that the perturbation-based method violates the assumption that the training and evaluation data come from the same distribution. In

response, Hooker et al. (2019) proposes a benchmark for evaluating feature importance estimates in DNNs. Their benchmark is called ROAR: **R**em**O**ve **A**nd **R**etrain. The goal of ROAR is to determine whether the removal of important information caused classification degradation or whether the introduction of the so-called uninformative information caused the modified images to go out-of-distribution, thereby causing classification degradation. It replaces the fraction of pixels deemed important according to some heatmap with the channel mean, similar to the perturbation-based method (Samek et al., 2016). However, there is an important difference: to deal with the fallacy of introducing out-of-distribution images in the evaluation phase, ROAR applies similar modifications to all the images in both training and test set. The explanation method is applied to all train and test set images to obtain a heatmap for each image. Then they remove the same percentage of the deemed important pixels from the image and replace it with the channel mean of that image. Finally, they train separate models on the modified data and evaluate the classification accuracy. If the accuracy of the re-trained models goes down, we can say with some certainty that the removed information in the modified image is indeed the cause of the classification degradation. Methods like Kindermans et al. (2019, 2016) investigate the reliability of heatmaps by modifying the input with information that does not change the classification result and checking how the heatmaps change as a result. They find that various visualization methods are vulnerable to input modification and return incorrect heatmaps as a result. The main conclusion is that many visualization methods are unreliable because they do not satisfy input invariance.

In contrast to previous methods, Vu et al. (2019) suggests a metric to evaluate heatmaps based on perturbing regions that are not indicated as important. Their metric is called c -Eval, where c is a number that indicates how robust the classifier is to perturbations in regions deemed as not important by the explanation method. This method indirectly measures how accurate the heatmaps are: the larger c , the more robust the classifier, the more accurate the explanation method is at identifying class-discriminating features. Using c -Eval they compare various explanation methods and find that there is a significant difference in the quality of heatmaps produced by black-box methods (e.g., SHAP, LIME) compared to back-propagation based methods (e.g., LRP, DeepLIFT).

In an alternative approach, Adebayo et al. (2018) proposes two sanity checks for evaluating the quality of heatmaps. The first is the *model parameter randomization test*, which compares heatmaps generated by a trained model with heatmaps generated by a randomly initialized model. If the outputs are similar, the explanation method is insensitive to model properties such as the weights. The second sanity check is the *data randomization test*, and it compares heatmaps generated by a model trained on the original dataset with heatmaps generated by a model trained on a version of the dataset where all the labels have been randomly permuted. If the heatmaps are similar, it indicates that the explanation method does not depend on the relationship between the data and the labels in the original data. The distance between the heatmaps is measured using various similarity metrics.

3.2.2 EVALUATING NLP EXPLANATIONS

The use of attention-based deep learning models for NLP has increased significantly (Vaswani et al., 2017; Brown et al., 2020). Attention has often been argued to be intrinsically inter-

pretable, see Section 2.3. However, recent studies (Jain & Wallace, 2019; Serrano & Smith, 2019; Baan et al., 2019) show that attention is not always interpretable and that attention does not always lead to insight into model prediction. Jain and Wallace (2019) first raises the point that, while we assume that attention is implicitly interpretable because directly it provides insight into which words are important, this assumption has never been formally evaluated. That is, the relationship between attention weights and model output is not clear. Their results show that the correlation between feature importance measures, like heatmapping, and the learned attention weights is weak. The results suggest that the ability of attention modules to provide meaningful explanations into model prediction is questionable at best. Serrano and Smith (2019) manipulates the attention weights in trained models and analyzes the resulting difference in model predictions. Their findings are mixed: sometimes higher attention weights correlate to model predictions but not always. Baan et al. (2019) reveals that some attention heads tend to specialize towards interpretable parts of a document, but this ability does not generalize to all documents. Also, the specializations are not consistent over differently initialized models.

To reconcile these conflicting views, some studies (Vashisht et al., 2019; Wiegreffe & Pinter, 2019) conduct experiments to find situations where attention can be used to gain insight into model prediction. Vashisht et al. (2019) presents experiments over a range of NLP tasks justifying both observations. They identify conditions when the attention weights are interpretable and correlate with text heatmaps. Their results also reveal that attention weights are not interpretable when the input only has a single sequence by showing that the attention weights function as a gating unit in this situation. Wiegreffe and Pinter (2019) provides a set of experiments that show that attention can be used to gain insight into model predictions. They conclude that the results from Jain and Wallace (2019) do not disprove that attention can serve as an explanation.

3.2.3 USING HUMANS TO EVALUATE EXPLANATIONS

Research in this explanation sub-field is still in its infancy and given that there is significant variation among people, contexts, and their needs, the results of the papers in this section should not be taken as absolute. In contrast to the previous evaluation methods, this section addresses methods concerned with how interpretable explanations are to humans. This approach benefits from revealing to what extent the setting and explanation method is interpretable and helpful to people who will use them. Figure 15 illustrates the various relationships that can exist between users and the model explanations. Research has revealed that there is still a big gap between the perceived and actual usefulness of explanations.

Model interpretability can be understood as how easy it is for a human to predict the model’s output on new input based on past predictions. This concept is called *simulability*. It was found that LIME improves the simulability of models trained on tabular data (Hase & Bansal, 2020). However, subjective ratings about the explanations did not predict how useful the explanations actually were.

Another way to judge explanations using a human baseline is by investigating how much model explanations align with human explanations. In an investigation of model alignment of transformer models for Natural Language Inference (NLI) it was found that BERT-based transformer models score the highest on alignment (Prasad et al., 2020). It should also be

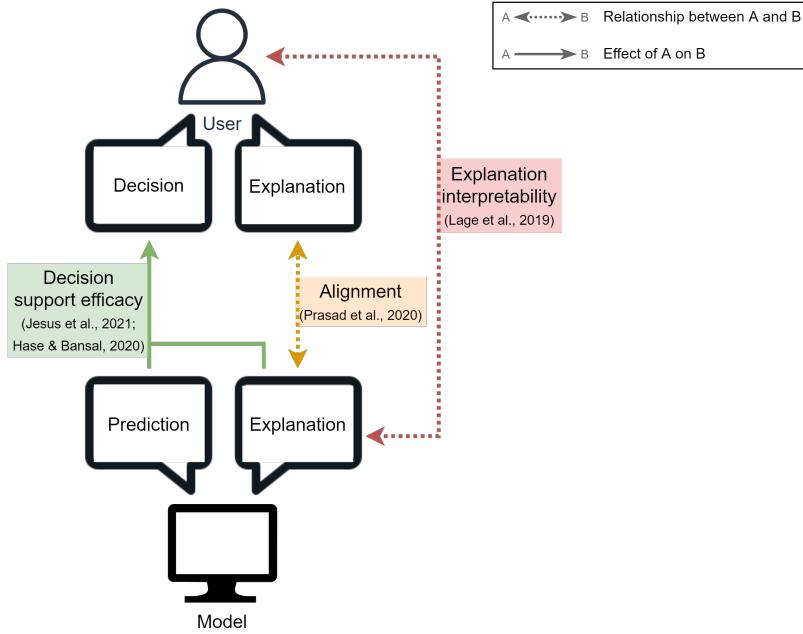


Figure 15: Current literature with a human in the loop evaluates XAI methods through the lens of the user experience and asks questions such as: What aspect of the explanation makes it interpretable for users? Or, to what extent are explanations actually useful to user decision-making? In reality, there are many relationships and aspects that should be taken into account when studying explanations through the lens of the user experience. This figure illustrates some of the relationships that have been studied in the current literature. The dotted arrows represent a relationship between aspects A and B, while a solid arrow represents the effect of aspect A on B. This figure is best viewed in color.

noted that the number of parameters in the model leads to worse model alignment and that alignment was not predicted by accuracy on NLI tasks.

Sometimes explanations like LIME and SHAP can hurt user performance, albeit not very much. In a recent study by Jesus et al. (2021) the accuracy and decision time was measured when participants needed to make decisions. The participants' accuracy was higher when only the basic data was given compared to when both the data and the explanation were given. However, the accuracy gap was not significant. The decision time significantly decreases when explanations are given.

An evaluation of which factors in explanations make them human interpretable concluded that explanations might have more in common with design principles (Lage et al., 2019). One factor that drove the results was the complexity of the explanations. The paper further identifies that regularizers can be used to optimize the interpretability of ML systems.

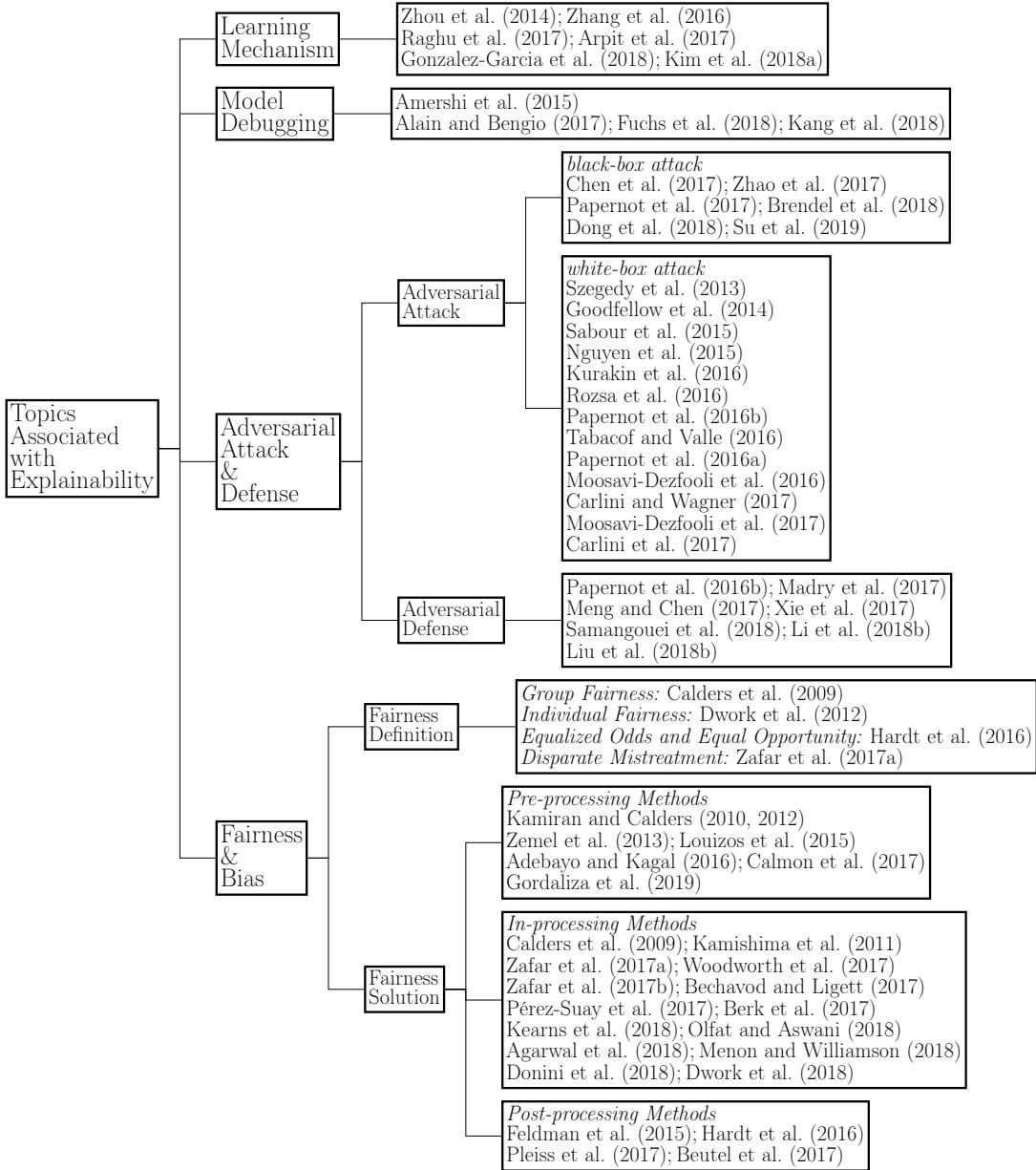


Figure 16: Topics associated with explainability.

4. Topics Associated with Explainability

We next review research topics closely aligned with explainable deep learning. A survey, visualized in Figure 16, identifies four broad related classes of research. Work on **learning mechanisms** (**Section 4.1**) investigates the backpropagation process to establish a theory around weight training. These studies, in some respects, try to establish a theory to explain how and why DNNs converge to some decision-making process. Research on **model debugging** (**Section 4.2**) develops tools to recognize and understand the failure modes

of a DNN. It emphasizes the discovery of problems that limit the training and inference process of a DNN (e.g., dead ReLUs, mode collapse, etc.). Techniques for **adversarial attack and defense (Section 4.3)** search for differences between regular and unexpected activation patterns. This line of work promotes deep learning systems that are robust and trustworthy; traits that also apply to explainability. Research on **fairness and bias in DNNs (Section 4.4)** is related to the ethics trait discussed above, but more narrowly concentrates on ensuring DNN decisions do not over-emphasize undesirable input data features. We elaborate on the connection between these research areas and explainable DNNs next.

4.1 Learning Mechanisms

The investigation of the learning mechanism tries to derive principles explaining the evolution of a model’s parameters during training. Many existing approaches can be categorized as being semantics-related, in that the analysis tries to associate a model’s learning process with concepts that have a concrete semantic meaning. They generally assign semantic concepts to a DNNs’ internal filters (weights) or representations (activations), in order to uncover a human-interpretable explanation of the learning mechanism. Semantically interpretable descriptions are rooted in the field of neuro-symbolic computing (Garcez et al., 2012). An early work is Zhou et al. (2014) which assigns semantic concepts, such as objects, object parts, etc, to the internal filters of a convolutional neural network (CNN) image scene classifier. Those semantic concepts are generated based on the visualization of receptive fields of each internal unit in the given layers. The authors also discovered that object detectors are embedded in a scene classifier without explicit object-level supervision for model training. Gonzalez-Garcia et al. (2018) further explores this problem in a quantitative fashion. Two quantitative evaluations are conducted to study whether the internal representations of CNNs really capture semantic concepts. Interestingly, the authors’ experimental results show that the association between internal filters and semantic concepts is modest and weak. But this association improves for deeper layers of the network, matching the conclusion of Zhou et al. (2014). Kim et al. (2018a) quantifies the importance of a given semantic concept with respect to a classification result via Testing with Concept Activation Vector (TCAV), which is based on multiple linear classifiers built with internal activations on prepared examples. The prepared examples contain both positive examples representing a semantic concept and randomly sampled negative examples that do not represent the concept. Directional derivatives are used to calculate TCAV, which measures the proportion of examples that belong to a given class that are positively influenced by a given concept.

Other methods to interpret the learning process of a DNN searches for statistical patterns indicative of convergence to a learned state. Those learning patterns include but are not limited to: i) how layers evolve along with the training process (Raghu et al., 2017); ii) the convergence of different layers (Raghu et al., 2017); and iii) the generalization and memorization properties of DNNs (Zhang et al., 2016; Arpit et al., 2017). In studying the learning dynamics during training, Raghu et al. (2017) makes a comparison between two different layers or networks via Singular Vector Canonical Correlation Analysis (SVCCA). For a neuron in a selected layer of a DNN, the neuron’s vector representation is generated in a “global fashion”, i.e. all examples from a given finite dataset are used, and each element in the neuron’s vector representation is an activation for an example. The vector representa-

tions for all neurons in a selected layer form a vector set, representing this layer. To compare two layers, SVCCA takes the vector set of each layer as input and calculates a canonical correlation similarity to make the alignment. The nature of SVCCA makes it a useful tool to monitor how layer activations evolve along with the training process. The authors further discover that earlier layers converge faster than later layers. Thus, the weights for earlier layers can be frozen earlier to reduce computational cost during training. Layer-wise convergence is also studied in work such as Zhang et al. (2016) using systematic experimentation. Keeping the model structure and hyper-parameters fixed, the authors' experiments are conducted only with different input modification settings, either on input labels or image pixels. The experimental results indicate that DNNs can perfectly fit training data with both random feature values and labels, while the degree of generalization on testing data reduces as randomness increases. The authors also hypothesize that explicit regularization (such as dropout, weight decay, data augmentation, etc.) *may* improve generalization and stochastic gradient descent could act as an implicit regularizer for linear models. In a similar study, Arpit et al. (2017) examine memorization by DNNs via quantitative experiments with real and random data. The study finds that DNNs do not simply memorize all real data; instead, patterns that are commonly shared among the data are leveraged for memorization. Interestingly, the authors claim that explicit regularization does make a difference in the speed of memorization for random data, which is different from the conclusions in Zhang et al. (2016). In a recent review paper Bahri et al. (2020) cover the intersection between statistical mechanics and deep learning and derives the success of deep learning from a theoretical perspective.

4.2 Model Debugging

The concept of model debugging applies techniques to find out when and where model architecture, data processing, and training-related errors occur. “Probes” may be leveraged to analyze the internal pattern of a DNN, to provide further hints towards performance improvement. A probe is an auxiliary model independent of the training process of the model (a DNN) that the probe investigates. Regardless of the form of the probe, the ultimate goal is model improvement. Kang et al. (2018) use *model assertions*, or Boolean functions, to verify the state of the model during training and run time. The assertions can be used to ensure model output is consistent with meta observations about the input. For example, if a model detects cars in a video, the cars should not disappear and reappear in successive frames of the video. Model debugging is thus implemented as a verification system surrounding the model and is implicitly model-agnostic. The model assertions are implemented as user-defined functions that operate on a recent model input and output history.

Researchers have explored several ways to use model assertions during both run-time and training time, correcting wrong outputs and collecting more samples to perform active learning. Amershi et al. (2015) proposes *ModelTracker*, a debugging framework revolving around an interactive visual interface. This visual interface summarizes traditional summary statistics, such as AUC and confusion matrices, and presents this summary to the user together with a visualization of how close data samples are to each other in the feature space. The interface also has an option to directly inspect prediction outliers in the form of the raw

data with its respective label, giving users the ability to correct mislabeled samples directly. This framework aims to provide a unified, model-agnostic inspection tool that supports debugging three specific types of errors: mislabeled data, inadequate features to distinguish between concepts, and insufficient data for generalizing from existing examples. Alain and Bengio (2017) use linear classifiers to understand the predictive power of representations learned by intermediate layers of a DNN. The features extracted by an intermediate layer of a deep classifier are fed as input to the linear classifier. The linear classifier has to predict which class the given input belongs to. The experimental results show that the performance of the linear classifier improves when making predictions using features from deeper layers, i.e., layers close to the final layer. This suggests that task-specific representations are encoded in the deeper layers.

Fuchs et al. (2018) proposes the idea of *neural stethoscopes*, which is a general-purpose framework used to analyze the DNN learning process by quantifying the importance of specific influential factors in the DNN and influence the DNN learning process by actively promoting and suppressing information. Neural stethoscopes extend a DNN's architecture with a parallel branch containing a two-layer perceptron. It is important to note that the main network branch does not need to be changed to be able to use the neural stethoscope. This parallel branch takes the feature representation from an arbitrary layer from the main network as input and is trained on a supplemental task given known complementary information about the dataset. Specifically, in this study the experiments are conducted on the ShapeStacks dataset (Groth et al., 2018), which introduces a vision-based stability prediction task for block towers. The dataset provides information on both the local and global stability of a stack of blocks. Here the stethoscope investigates the internal representations in the network layers that lead to predicting the global stability of a stack of blocks, with local stability as complementary information. The stethoscope can be tuned to three different modes of operation: analytic, auxiliary, and adversarial. Each mode determines how the stethoscope loss L_S is propagated, e.g., in the analytical mode, L_S is not propagated through the main network. The auxiliary and adversarial modes are used to promote and suppress information, respectively. The paper shows that the method successfully improved network performance and mitigated biases that are present in the dataset.

4.3 Adversarial Attack and Defense

An adversarial example is an artificial input engineered to disturb the judgment of a DNN intentionally (Goodfellow et al., 2014). Developing defenses to adversarial examples requires a basic understanding of the space that inputs are taken from and the shape and form of boundaries between classes. Interpretations of this space inform the construction of defenses to better discriminate between classes and form the basis of explaining input/output behavior. Moreover, an "explanation" from a model that is not reasonable given its input and output may indicate an adversarial example.

The study of adversarial examples (Yuan et al., 2019; Zhang et al., 2019b) are from the perspective of attack and defense. *Adversarial attack* methods are about generating adversarial examples that can fool a DNN. From the model access perspective, there are two main types of adversarial attack: *black-box* (Chen et al., 2017; Zhao et al., 2017; Papernot et al., 2017; Brendel et al., 2018; Dong et al., 2018; Su et al., 2019) and *white-box* (Szegedy

et al., 2013; Goodfellow et al., 2014; Sabour et al., 2015; Nguyen et al., 2015; Kurakin et al., 2016; Rozsa et al., 2016; Papernot et al., 2016a; Moosavi-Dezfooli et al., 2016; Tabacof & Valle, 2016; Kurakin et al., 2016; Carlini & Wagner, 2017; Moosavi-Dezfooli et al., 2017; Carlini et al., 2017; Eykholt et al., 2018) attacks. In the black-box setting the attacker has no access to the model parameters or intermediate gradients whereas these are available for the white-box settings. *Adversarial defense* (Madry et al., 2017; Papernot et al., 2016b; Meng & Chen, 2017; Xie et al., 2017; Samangouei et al., 2018; Li et al., 2018b; Liu et al., 2018b), on the other hand, seeks solutions to make a DNN robust against generated adversarial examples.

Recent work on adversarial attacks reveals vulnerabilities by perturbing input data with imperceptible noise (Goodfellow et al., 2014; Carlini & Wagner, 2017; Madry et al., 2017) or by adding “physical perturbations” to objects under analysis (i.e., black and white stickers on objects captured by computer vision systems) (Eykholt et al., 2018). Among numerous adversarial attack methods, the C&W attack (Carlini & Wagner, 2017) and Projected Gradient Descent (PGD) attack (Madry et al., 2017) are frequently used to evaluate the robustness of DNNs. The C&W attack (Carlini & Wagner, 2017) casts the adversarial attack task as an optimization problem and is originally proposed to challenge an adversarial defense method called defensive distillation (Papernot et al., 2016b). Variants of C&W attacks are based on the distance metrics (ℓ_0 , ℓ_2 , or ℓ_∞). Carlini and Wagner (2017), for example, can successfully defeat defensive distillation with high-confidence adversarial examples generated via C&W attack. The PGD attack (Madry et al., 2017) is an iterative version of an early stage adversarial attack called Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014). As indicated in its name, PGD attacks generate adversarial examples based on the gradients of the loss with respect to the input. A PGD attack is more favorable than a C&W attack when direct control of input distortion is needed (Liu et al., 2018b).

Adversarial defense is challenging due to the diversity of the adversarial example crafting processes and a DNN’s high-dimensional feature space. There exist two typical groups of adversarial defense methods, i) adversarial training (Madry et al., 2017; Goodfellow et al., 2014; Szegedy et al., 2013), which is to augment the training dataset with generated adversarial examples such that the trained model is more robust against adversarial attacks, and ii) removal perturbations (Samangouei et al., 2018; Meng & Chen, 2017), which dismisses adversarial perturbations from input data. Madry et al. (2017) integrates the PGD attack into the model training process, such that the model is optimized on both benign examples and challenging adversarial examples. The optimization is conducted in a min-max fashion, where the loss for adversarial attack process is maximized in order to generate strong adversarial examples, while the loss for the classification process is minimized in order to get a robust and well-performed model. Samangouei et al. (2018), on the other hand, tackles the adversarial defense problem by filtering out adversarial perturbations. Generative Adversarial Networks (GANs) are leveraged to project a given input image, potentially polluted by adversarial perturbations, into a pseudo-original image, where adversarial artifacts are diminished. Model decisions are made from the GAN-generated “original” image. Experiments indicate that this defense technique is effective against both black-box and white-box attacks.

4.4 Fairness and Bias

Model fairness aims to build DNN models that objectively consider each input feature and is not unduly biased against a particular subset of the input data. Although a firm definition of what it means for a DNN to be “fair” is evolving, common themes are emerging in the literature (Heidari et al., 2018). *Group fairness* (Calders et al., 2009), also called *demographic parity* or *statistical parity*, focuses on fairness with respect to a group (based on race, gender, etc.). The goal of group fairness is to ensure each group receives equalized percentage of benefit. Consider a loan application as an example. Suppose we are monitoring the loan approval situation of two cities, city A and city B. The population of city A is twice as much as that of city B. Based on the definition of group fairness, twice as many loan applications should be approved in A compared to city B. *Individual fairness* (Dwork et al., 2012) aims to treat similar inputs similarly based on a metric to measure the closeness of their features. To compare group fairness and individual fairness, consider the loan request example again. Under the restriction of group fairness, an individual from city A may not be approved for a loan request just because of the group percentage limitation, even though this individual is more qualified based on economic metrics than other approved ones from city B. However, individual fairness requires that individuals with similar characteristics have the same chance to be approved for a loan request, regardless of which city individuals come from. This is in antithesis with group fairness. Further notions of fairness, such as *equalized odds* and *equal opportunity* (Hardt et al., 2016), *disparate mistreatment* (Zafar et al., 2017a), and others (Heidari et al., 2018; Woodworth et al., 2017) are also studied in the literature.

The fairness problem is currently addressed by three types of methods (Calmon et al., 2017): (i) *pre-processing* methods revise input data to remove information correlated to sensitive attributes; (ii) *in-process* methods add fairness constraints into the model learning process; and (iii) *post-process* methods adjust model predictions after the model is trained. *Pre-processing* methods (Kamiran & Calders, 2010, 2012; Zemel et al., 2013; Louizos et al., 2015; Adebayo & Kagal, 2016; Calmon et al., 2017; Gordaliza et al., 2019) learn an alternative representation of the input data that removes information correlated to the sensitive attributes (such as race or gender) while maintaining the model performance as much as possible. For example, Calmon et al. (2017) proposes a probabilistic framework to transform input data to prevent unfairness in the scope of supervised learning. The input transformation is conducted as an optimization problem, aiming to balance discrimination control (group fairness), individual distortion (individual fairness), and data utility. *In-process* methods (Calders et al., 2009; Kamishima et al., 2011; Zafar et al., 2017a; Woodworth et al., 2017; Zafar et al., 2017b; Bechavod & Ligett, 2017; Kearns et al., 2018; Pérez-Suay et al., 2017; Berk et al., 2017; Olfat & Aswani, 2018; Agarwal et al., 2018; Menon & Williamson, 2018; Donini et al., 2018; Dwork et al., 2018) directly introduce fairness learning constraints to the model in order to punish unfair decisions during training. Kamishima et al. (2011) achieve the fairness goal by adding a fairness regularizer such that the influence of sensitive information on model decisions is reduced. *Post-process* methods (Feldman et al., 2015; Hardt et al., 2016; Pleiss et al., 2017; Beutel et al., 2017) are characterized by adding ad-hoc fairness procedures to a trained model. One example is Hardt et al. (2016) which constructs non-discriminating predictors as a post-processing step to achieve equalized odds and equal opportunity (two fairness notions proposed in their study). They introduce the procedure

to construct non-discriminating predictors for two scenarios of the original model, binary predictor and score function, wherein the latter scenario, the original model generates real score values in range [0, 1]. A non-discriminating predictor is constructed for each protected group, with a defined threshold to achieve a fairness goal.

5. Designing Explanations for Users

The foundations of explaining DNNs discussed in this survey are seldom enough to achieve explanations helpful to users in practice. ML engineers designing explainable DNNs must often integrate an explanatory method into their DNN and then refine the presentation of the explanation to a form useful for the end-user. A *useful* explanation must conform to some definition of what constitutes a satisfactory explanation of the network’s inner workings depending on the user, the conditions of use, and the task at hand. These definitions are often qualitative (e.g., one user is better swayed by visual over textual explanations for a task). User requirements for an explanation may further vary by preferences between high fidelity explanations versus those that are parsimonious. The quality of an explanation depends on the user- and context-specific utility and makes the evaluation of explanations a difficult problem. This suggests that explanations, grounded in the methods discussed in this field guide, need to be designed by engineers on a case-by-case basis for the user and task at hand. This section describes important design questions that should be considered when applying the methods in this field guide in practice:

1. **Who is the end-user?** The kind of end-user, and in particular their expertise in deep learning and their domain-specific requirements, define the appropriate trade-off between fidelity and parsimony in an explanation’s presentation.
2. **How practically impactful are the decisions of the DNN?** Here impact corresponds to the consequence of right and wrong decisions on people and society. Time-critical scenarios require explanations that can be rapidly generated and processed by a user should there be a need to intervene (e.g., in self-driving cars). Decision-critical scenarios require explanations that are *trustworthy*, that is, an explanation that a user trusts to be faithful to the actual decision-making process of the DNN.
3. **How extendable is an explanation?** It is expensive to design a form of explanation for only a single type of user who faces a single type of problem. A good design should be grounded on a single user’s preferences, but that can be applied to multiple types of problems or be flexible enough to appeal to multiple user types examining the same problem type. It may not be feasible to devise the presentation of an explanation that appeals to a broad set of users tailored to a diverse set of problems.

5.1 Understanding the End User

One of the primary tasks to design an explanation is to determine the type of end-user using the system. The literature has documented cases of designs that provide both low-level technical specific explanations targeting on deep learning experts (Zeiler & Fergus, 2014; Sundararajan et al., 2017; Anderson et al., 2018; Li et al., 2016; Fong & Vedaldi, 2017; Zintgraf et al., 2017), and high-level reasoning extracted explanations catering to normal

users (Harradon et al., 2018; Zhang et al., 2019a, 2017, 2018). DNN experts care mostly about technical details and potential hints for model revising and performance improvement. Ideal explanations for them could be in form of input feature influence analytics (Adler et al., 2018; Koh & Liang, 2017; Li et al., 2016; Fong & Vedaldi, 2017), hidden states interaction and visualizations (Anderson et al., 2018; Vaswani et al., 2017; Vinyals et al., 2015; Zeiler & Fergus, 2014; Selvaraju et al., 2017; Bach et al., 2015; Sundararajan et al., 2017), etc. DNN experts, for instance, could check if the model is emphasizing reasonable image areas (Zeiler & Fergus, 2014) or text elements/words (Vaswani et al., 2017; Sundararajan et al., 2017) towards generating corresponding model decisions and propose model revision strategies accordingly. On the other hand, normal users mainly focus on the high-level functionality of the model instead of technical details. Their main concern is if the model is working reasonably and not violating human logic. The explanation can be represented in the form of extracted reasoning logic (Harradon et al., 2018; Zhang et al., 2019a, 2017, 2018) or some easy to understandable input clues with respect to given prediction (Ribeiro et al., 2016c). If the decision is generated from unexpected input elements or does not follow a logical reasoning process, the user could doubt and deny the model decision. Considering the different user expertise levels on DNN knowledge, designing a general model explanation system, which satisfies both DNN experts and normal users, is challenging and remains to be explored.

The domain a user operates in is another important consideration. For example, the explanation needs of a medical doctor require that the explanation representation be detailed enough such that the doctor can understand the reasoning process behind the specific diagnosis and be confident about said diagnosis (Lipton, 2017), e.g., the patient needs this specific treatment because it identifies features of cancer at a particular stage. But, no explainable method can automatically tailor its explanations to end-users for a specific domain. One way to obtain such explanations using current methods is if the features of the input data expressed by an explanation method have an intuitive domain-specific interpretation built upon a systematic knowledge base constructed by domain experts.

5.2 The Impact of DNN Decisions

The need for an explanation and its properties depend on the impact of a DNN’s operation on human life and society. This impact can be realized based on the result and speed of a decision. In *time-critical* scenarios (Grigorescu et al., 2020) where users must process and react to DNN decisions in limited time, explanations must be produced that are simple to interpret and understand and are not computationally intense to perform. This is a crucial aspect of explanations that is seldom investigated in the literature. For example, a DNN providing recommendations during a military operation, or sensing upcoming hazards to a driven vehicle, needs to support their output with explanations while giving the user enough time to process and react accordingly. In a *decision-critical* scenario (Grigorescu et al., 2020; Nemati et al., 2018; Ahmad et al., 2018), the ability to not only interpret but deeply inspect a decision grows in importance. Any user decision based on a DNN’s recommendation should be supported with evidence and explanations others can understand. At the same time, should a DNN’s recommendation turn out to be incorrect or lead to an undesirable outcome for the user, the model should be inspected post-hoc to hypothesize

root causes and identify “bugs” in the DNN’s actions. Deep, technical inspections of the neural network guided by comprehensive interpretations of its inference and training actions are necessary for such post-hoc analysis. Few current model explanations are designed with time- and decision-critical scenarios in mind. The computational cost of many model explanations tends to be high and may require additional human labor, which is undesirable if an automatic and instant explanation is needed. For instance, for explanations presented in form of model visualization (Zeiler & Fergus, 2014; Selvaraju et al., 2017; Shrikumar et al., 2017; Sundararajan et al., 2017; Montavon et al., 2017; Zhou et al., 2016), additional human effort is needed for verification, which is potentially costly. Some explanation methods with post-hoc training involved (Ribeiro et al., 2016c; Frosst & Hinton, 2017; Krakovna & Doshi-Velez, 2016; Hou & Zhou, 2020) may be limited in their utility in providing explanations for real-time input. The study for decision-critical scenarios is still under development. In order to increase the fidelity and reliability of model decisions, a variety of topics are explored besides model explanations, including model robustness (Papernot et al., 2016b; Meng & Chen, 2017; Xie et al., 2017; Samangouei et al., 2018; Li et al., 2018b), fairness and bias (Heidari et al., 2018; Calders et al., 2009; Hardt et al., 2016; Zafar et al., 2017a; Calmon et al., 2017; Gordaliza et al., 2019; Agarwal et al., 2018; Menon & Williamson, 2018; Donini et al., 2018; Dwork et al., 2018; Pleiss et al., 2017; Beutel et al., 2017) and model trustworthiness (Jiang et al., 2018; Heo et al., 2018). The study of the topics mentioned earlier, together with model explanations, may shed light on potential new solutions for applications in decision-critical scenarios.

5.3 Design Extendability

Modularity and reusability are important extendability traits in the architecture of large-scale software systems: modularity promotes the ability of an engineer to replace and alter system components as necessary, while reusability promotes the use of already proven software modules. In a similar vein, highly reliable and performant DNN systems should also be constructed with reusable and highly modular components. Modularity is a trait of the functional units of a DNN that may be adaptable for multiple architectures, such as the form of an attention mechanism suitable for sequential data processing (Vaswani et al., 2017; Devlin et al., 2019). A highly modular DNN architecture may contain many “plug and play” components in each layer so that its complete design can be seen as a composition of interconnected functional units. Reusability applies to complete DNN systems, perhaps already trained, that can be reused in multiple problem domains. One example of reusability is the common application of a pre-trained YOLO (Redmon et al., 2016) model for object localization in frames in a deep learning video processing pipeline.

DNN explanation methods that exhibit these extendability traits are likely to be useful over various DNN models and application domains. Modularized explainable models will crucially reduce the overhead in implementing and deploying explainability in new domains and may lead to explanatory forms a user is familiar with over multiple types of models. Reusability plays a role in risk control, such that the fidelity of an explanation remains consistent however the explanatory method is applied.

Neither modularity nor reusability is the focus of explainable methods in the literature. However, existing methods could be divided by how modular they potentially are. Model-

agnostic methods (Ribeiro et al., 2016c, 2016b, 2016a; Fong & Vedaldi, 2017; Jha et al., 2017), which do not take the type of model into account, are modular by definition in the sense that the explanatory module is independent of the model it is producing explanations for. On the other hand, the second category contains explanation methods that are specific to the model (Shrikumar et al., 2017; Zeiler & Fergus, 2014; Bach et al., 2015; Montavon et al., 2017; Zhou et al., 2016; Murdoch et al., 2018; Xie et al., 2017; Park et al., 2018; Hendricks et al., 2016). This aspect is important for expert users that are developing deep learning models and need to understand specifically which aspect of the deep learning model is influencing the predictions, e.g., in model debugging. However, these methods, by their very nature, lack modularity.

6. Future Directions

This field guide concludes by introducing research directions whose developments can contribute to improving explainable deep learning.

A Unifying Approach to Explainability. There have been several efforts to come up with a framework for explainable artificial intelligence (XAI) or interpretable machine learning (Gilpin et al., 2018; Doshi-Velez & Kim, 2017, 2018). Existing works consider efforts from different perspectives. There remains a lack of systematic general theory in the realm of DNN explanation (Arrieta et al., 2020; Díez et al., 2013). A systematic theory should benefit the overall model explanation studies, and once formed, some current challenging explainable problems may be adequately handled, and some novel directions may be proposed based on the systematic theories. However, one of the main reasons why it is so difficult to establish a formal theory of explanation is that the basic concepts of explanations in AI are difficult or impossible to formalize (Wolf et al., 2019).

User-friendly Explanations. User-friendly explanations are needed to minimize the technical understanding needed by a user to interpret explanations correctly. As the concern of the opaque nature of DNNs raises more attention in society, model explanations may become mandatory in a wide range of real-life applications (Goodman & Flaxman, 2017). Given the varied backgrounds of model users, explanation friendliness may be a future trend to construct explanations of high quality. Most explainable methods still cater towards expert users instead of laymen (Ras et al., 2018), in the sense that knowledge about the method is needed to understand the explanation. The requirement on model knowledge limits the wide usage of such explanation models since, in real scenarios, the chance of the end-users being machine learning experts is very low. Assuming the end-user has been correctly determined, the next step is to determine *what* aspect of a model needs explaining.

Producing Explanations Efficiently. Time and decision-critical explanations (Grigorescu et al., 2020; Ahmad et al., 2018; Nemati et al., 2018), as discussed in Section 5.2 must be produced with enough time for a user to react to a DNN’s decision. An efficient manner to produce explanations saves computational power, which is favorable in industrial applications or when explanations are required in environments with low computing

resources.

Developing Methods for Trustworthiness. The vulnerability of a DNN to adversarial examples (Yuan et al., 2019; Goodfellow et al., 2014; Carlini & Wagner, 2017; Madry et al., 2017) and poisoned training sets (Saha et al., 2020) raises much concern on trustworthiness. As more and more DNNs are leveraged in real-life applications, the demand for model trustworthiness would undoubtedly increase, especially for decision-critical scenarios where undesired decisions may cost severe consequences. This thread of research is only beginning to be developed (Jiang et al., 2018; Heo et al., 2018).

7. Conclusions

The rapid advancements in deep neural networks have stimulated innovations in a wide range of applications from facial recognition (Masi et al., 2018) and explainable deep learning, voice assistance (Tulshan & Dhage, 2018), to self-driving vehicle systems (Jain et al., 2015). The field is motivated by the opaque nature of DNN systems and the increasing demand for model transparency and trustworthiness in society. Government policies, such as the EU’s General Data Protection Regulation (GDPR) (Goodman & Flaxman, 2017), allude to a future where the explainability aspects of deep networks will become a legal concern.

We hope this field guide has distilled the essential topics, related work, methods, and concerns associated with explainable deep learning for an initiate. A wide range of existing methods on deep learning explainability is introduced and organized by a novel categorization scheme to depict the field. Topics closely associated with DNN explainability, including model learning mechanisms, model debugging, adversarial attack and defense, and model fairness and bias are reviewed as related work. A discussion on user-oriented explanation designing and future trends of this field is provided at the end of this survey, shedding light on potential directions on model explainability. Given the countless papers in this field and the rapid development of explainable methods, we admit the field guide cannot cover every paper or every aspect that belongs to this realm.

In the end, the important thing is to explain the right thing to the right person in the right way at the right time.¹ We are excited to continue to observe how the field evolves to deliver the appropriate explanation to the right audience who need it the most. We hope the numerous solutions actively being explored will lead to the fairer, safer, and more confident use of deep learning across society.

Acknowledgments

Gabriëlle Ras and Ning Xie are co-first authors on this work. Ning Xie and Derek Doran completed some of this work while at the Dept. of Computer Science and Engineering, Wright State University, Dayton, OH. We acknowledge project support from the Ohio Federal Research Network, the Multidisciplinary Research Program of the Department of Defense (MURI N00014-00-1-0637), and the organizers and participants of the Schloss Dagstuhl – Leibniz Center for Informatics Seminar 17192 on Human-Like Neural-Symbolic Comput-

1. Paraphrased from Dr. Silja Renooij at BENELEARN2019

ing for providing the environment to develop the ideas in this paper. Parts of this work was completed under a Fulbright-NSF Fellowship for Cyber Security and Critical Infrastructure. We would also like to thank Erdi Çallı and Pim Haselager for the helpful discussions and general support.

References

- Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Süstrunk, S. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11), 2274–2282.
- Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, 6, 52138–52160.
- Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., & Kim, B. (2018). Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, pp. 9525–9536.
- Adebayo, J., & Kagal, L. (2016). Iterative orthogonal feature projection for diagnosing bias in black-box models. *ArXiv*, *abs/1611.04967*.
- Adler, P., Falk, C., Friedler, S. A., Nix, T., Rybeck, G., Scheidegger, C., Smith, B., & Venkatasubramanian, S. (2018). Auditing black-box models for indirect influence. *Knowledge and Information Systems*, 54(1), 95–122.
- Agarwal, A., Beygelzimer, A., Dudik, M., Langford, J., & Wallach, H. (2018). A reductions approach to fair classification. In *International Conference on Machine Learning*, pp. 60–69.
- Ahmad, M. A., Eckert, C., & Teredesai, A. (2018). Interpretable machine learning in health-care. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pp. 559–560.
- Alain, G., & Bengio, Y. (2017). Understanding intermediate layers using linear classifier probes. In *International Conference on Learning Representations*.
- Alvarez-Melis, D., & Jaakkola, T. S. (2018). On the robustness of interpretability methods. *ArXiv*, *abs/1806.08049*.
- Amershi, S., Chickering, M., Drucker, S. M., Lee, B., Simard, P., & Suh, J. (2015). Model-tracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 337–346.
- Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., & Zhang, L. (2018). Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6077–6086.
- Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Lawrence Zitnick, C., & Parikh, D. (2015). VQA: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2425–2433.

- Arpit, D., Jastrzębski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., et al. (2017). A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pp. 233–242.
- Arras, L., Horn, F., Montavon, G., Müller, K.-R., & Samek, W. (2016). Explaining predictions of non-Linear classifiers in NLP. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pp. 1–7.
- Arras, L., Horn, F., Montavon, G., Müller, K.-R., & Samek, W. (2017). "What is relevant in a text document?": An interpretable machine learning approach. *PLOS One*, 12.
- Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al. (2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58, 82–115.
- Baan, J., ter Hoeve, M., van der Wees, M., Schuth, A., & de Rijke, M. (2019). Do transformer attention heads provide transparency in abstractive summarization?. *ArXiv*, *abs/1907.00570*.
- Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., & Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS One*, 10.
- Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., & Müller, K.-R. (2010). How to explain individual classification decisions. *Journal of Machine Learning Research*, 11, 1803–1831.
- Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.
- Bahri, Y., Kadmon, J., Pennington, J., Schoenholz, S. S., Sohl-Dickstein, J., & Ganguli, S. (2020). Statistical mechanics of deep learning. *Annual Review of Condensed Matter Physics*, 11, 501–528.
- Bastani, O., Kim, C., & Bastani, H. (2017). Interpreting blackbox models via model extraction. *ArXiv*, *abs/1705.08504*.
- Bechavod, Y., & Ligett, K. (2017). Penalizing unfairness in binary classification. *ArXiv*, *abs/1707.00044*.
- Berk, R., Heidari, H., Jabbari, S., Joseph, M., Kearns, M., Morgenstern, J., Neel, S., & Roth, A. (2017). A convex framework for fair regression. *ArXiv*, *abs/1706.02409*.
- Beutel, A., Chen, J., Zhao, Z., & Chi, E. H. (2017). Data decisions and theoretical implications when adversarially learning fair representations. *ArXiv*, *abs/1707.00075*.
- Bien, J., & Tibshirani, R. (2011). Prototype selection for interpretable classification. *The Annals of Applied Statistics*, 5, 2403–2424.
- Brendel, W., Rauber, J., & Bethge, M. (2018). Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representations*.

- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, pp. 1877–1901.
- Buolamwini, J., & Gebru, T. (2018). Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on Fairness, Accountability and Transparency*, pp. 77–91.
- Calders, T., Kamiran, F., & Pechenizkiy, M. (2009). Building classifiers with independency constraints. In *2009 IEEE International Conference on Data Mining Workshops*, pp. 13–18.
- Calmon, F., Wei, D., Vinzamuri, B., Ramamurthy, K. N., & Varshney, K. R. (2017). Optimized pre-processing for discrimination prevention. In *Advances in Neural Information Processing Systems*, pp. 3992–4001.
- Camburu, O.-M., Rocktäschel, T., Lukasiewicz, T., & Blunsom, P. (2018). e-SNLI: Natural language inference with natural language explanations. In *Advances in Neural Information Processing Systems*, pp. 9560–9572.
- Carlini, N., Katz, G., Barrett, C., & Dill, D. L. (2017). Ground-truth adversarial examples. *ArXiv*, *abs/1709.10207*.
- Carlini, N., & Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy*, pp. 39–57.
- Carter, S., Armstrong, Z., Schubert, L., Johnson, I., & Olah, C. (2019). Activation atlas. *Distill*, *4*(3).
- Carvalho, D. V., Pereira, E. M., & Cardoso, J. S. (2019). Machine learning interpretability: A survey on methods and metrics. *Electronics*, *8*(8), 832.
- Chen, C., Li, O., Tao, D., Barnett, A., Rudin, C., & Su, J. K. (2019). This looks like that: Deep learning for interpretable image recognition. In *Advances in Neural Information Processing Systems*, pp. 8928–8939.
- Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., & Hsieh, C.-J. (2017). ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 15–26.
- Chong, E., Han, C., & Park, F. C. (2017). Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, *83*, 187–205.
- Craven, M., & Shavlik, J. W. (1996). Extracting tree-structured representations of trained networks. In *Advances in Neural Information Processing Systems*, pp. 24–30.
- Dabkowski, P., & Gal, Y. (2017). Real time image saliency for black box classifiers. In *Advances in Neural Information Processing Systems*, pp. 6967–6976.
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4171–4186.

- DeYoung, J., Jain, S., Rajani, N. F., Lehman, E., Xiong, C., Socher, R., & Wallace, B. C. (2020). ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4443–4458.
- Díez, J., Khalifa, K., & Leuridan, B. (2013). General theories of explanation: Buyer beware. *Synthese*, 190(3), 379–396.
- Ding, Y., Liu, Y., Luan, H., & Sun, M. (2017). Visualizing and understanding neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vol. 1, pp. 1150–1159.
- Dong, W., Li, J., Yao, R., Li, C., Yuan, T., & Wang, L. (2016). Characterizing driving styles with deep learning. *ArXiv*, *abs/1607.03611*.
- Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., & Li, J. (2018). Boosting adversarial attacks with momentum. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9185–9193.
- Dong, Y., Su, H., Zhu, J., & Zhang, B. (2017). Improving interpretability of deep neural networks with semantic information. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4306–4314.
- Donini, M., Oneto, L., Ben-David, S., Shawe-Taylor, J. S., & Pontil, M. (2018). Empirical risk minimization under fairness constraints. In *Advances in Neural Information Processing Systems*, pp. 2791–2801.
- Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *ArXiv*, *abs/1702.08608*.
- Doshi-Velez, F., & Kim, B. (2018). Considerations for evaluation and generalization in interpretable machine learning. In *Explainable and Interpretable Models in Computer Vision and Machine Learning*, pp. 3–17. Springer.
- Došilović, F. K., Brčić, M., & Hlupić, N. (2018). Explainable artificial intelligence: A survey. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics*, pp. 0210–0215.
- Dwork, C., Hardt, M., Pitassi, T., Reingold, O., & Zemel, R. (2012). Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pp. 214–226.
- Dwork, C., Immorlica, N., Kalai, A. T., & Leiserson, M. (2018). Decoupled classifiers for group-fair and efficient machine learning. In *Conference on Fairness, Accountability and Transparency*, pp. 119–133.
- Elenberg, E., Dimakis, A. G., Feldman, M., & Karbasi, A. (2017). Streaming weak submodularity: Interpreting neural networks on the fly. In *Advances in Neural Information Processing Systems*, pp. 4044–4054.
- Erhan, D., Bengio, Y., Courville, A., & Vincent, P. (2009). Visualizing higher-layer features of a deep network. *Département d'Informatique et Recherche Opérationnelle, University of Montreal, QC, Canada, Tech. Rep*, 1341.

- Erhan, D., Courville, A., & Bengio, Y. (2010). Understanding representations learned in deep architectures. *Département d'Informatique et Recherche Opérationnelle, University of Montreal, QC, Canada, Tech. Rep, 1355*.
- Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature, 542*(7639), 115–118.
- Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., & Song, D. (2018). Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1625–1634.
- Feldman, M., Friedler, S. A., Moeller, J., Scheidegger, C., & Venkatasubramanian, S. (2015). Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 259–268.
- Fong, R. C., & Vedaldi, A. (2017). Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3429–3437.
- Frosst, N., & Hinton, G. (2017). Distilling a neural network into a soft decision tree. *ArXiv, abs/1711.09784*.
- Fuchs, F. B., Groth, O., Kosoriek, A. R., Bewley, A., Wulfmeier, M., Vedaldi, A., & Posner, I. (2018). Neural stethoscopes: Unifying analytic, auxiliary and adversarial network probing. *ArXiv, abs/1806.05502*.
- Garcez, A. S. d., Broda, K. B., & Gabbay, D. M. (2012). *Neural-symbolic learning systems: Foundations and applications*. Springer Science & Business Media.
- Garvie, C. (2016). *The perpetual line-up: Unregulated police face recognition in America*. Georgetown Law, Center on Privacy & Technology.
- Georgiev, P., Bhattacharya, S., Lane, N. D., & Mascolo, C. (2017). Low-resource multi-task audio sensing for mobile and embedded devices via shared deep neural network representations. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 1*(3), 50.
- Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., & Kagal, L. (2018). Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics*, pp. 80–89.
- Gonzalez-Garcia, A., Modolo, D., & Ferrari, V. (2018). Do semantic parts emerge in convolutional neural networks?. *International Journal of Computer Vision, 126*(5), 476–494.
- Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.
- Goodman, B., & Flaxman, S. (2017). European Union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine, 38*(3), 50–57.
- Gordaliza, P., Del Barrio, E., Fabrice, G., & Loubes, J.-M. (2019). Obtaining fairness using optimal transport theory. In *International Conference on Machine Learning*, pp. 2357–2365.

- Goswami, G., Bhardwaj, R., Singh, R., & Vatsa, M. (2014). MDLFace: Memorability augmented deep learning for video face recognition. In *IEEE International Joint Conference on Biometrics*, pp. 1–7.
- Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., & Parikh, D. (2017). Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6904–6913.
- Graves, A., Wayne, G., & Danihelka, I. (2014). Neural turing machines. *ArXiv*, *abs/1410.5401*.
- Grigorescu, S., Trasnea, B., Cocias, T., & Macesanu, G. (2020). A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, *37*(3), 362–386.
- Groth, O., Fuchs, F. B., Posner, I., & Vedaldi, A. (2018). ShapeStacks: Learning vision-based physical intuition for generalised object stacking. In *Proceedings of the European Conference on Computer Vision*, pp. 702–717.
- Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., & Pedreschi, D. (2018). A survey of methods for explaining black box models. *ACM Computing Surveys*, *51*(5), 93.
- Hardt, M., Price, E., Srebro, N., et al. (2016). Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems*, pp. 3315–3323.
- Harradon, M., Druce, J., & Ruttenberg, B. (2018). Causal learning and explanation of deep neural networks via autoencoded activations. *ArXiv*, *abs/1802.00541*.
- Hase, P., & Bansal, M. (2020). Evaluating explainable AI: Which algorithmic explanations help users predict model behavior?. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5540–5552.
- He, R., Lee, W. S., Ng, H. T., & Dahlmeier, D. (2018). Effective attention modeling for aspect-level sentiment classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 1121–1131.
- Heidari, H., Ferrari, C., Gummadi, K., & Krause, A. (2018). Fairness behind a veil of ignorance: A welfare analysis for automated decision making. In *Advances in Neural Information Processing Systems*, pp. 1265–1276.
- Hendricks, L. A., Akata, Z., Rohrbach, M., Donahue, J., Schiele, B., & Darrell, T. (2016). Generating visual explanations. In *European Conference on Computer Vision*, pp. 3–19.
- Heo, J., Lee, H. B., Kim, S., Lee, J., Kim, K. J., Yang, E., & Hwang, S. J. (2018). Uncertainty-aware attention for reliable interpretation and prediction. In *Advances in Neural Information Processing Systems*, pp. 909–918.
- Heskes, T., Sijben, E., Bucur, I. G., & Claassen, T. (2020). Causal Shapley values: Exploiting causal knowledge to explain individual predictions of complex models. In *Advances in Neural Information Processing Systems*, pp. 4778–4789.
- Hind, M., Wei, D., Campbell, M., Codella, N. C., Dhurandhar, A., Mojsilović, A., Natesan Ramamurthy, K., & Varshney, K. R. (2019). TED: Teaching AI to explain its

- decisions. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 123–129.
- Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *ArXiv, abs/1503.02531*.
- Hooker, G. (2004). Discovering additive structure in black box functions. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 575–580.
- Hooker, G. (2007). Generalized functional anova diagnostics for high-dimensional functions of dependent variables. *Journal of Computational and Graphical Statistics, 16*(3), 709–732.
- Hooker, S., Erhan, D., Kindermans, P.-J., & Kim, B. (2019). A benchmark for interpretability methods in deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 9734–9745.
- Hoos, H., & Leyton-Brown, K. (2014). An efficient approach for assessing hyperparameter importance. In *International Conference on Machine Learning*, pp. 754–762.
- Hou, B.-J., & Zhou, Z.-H. (2020). Learning with interpretable structure from gated RNN. *IEEE Transactions on Neural Networks and Learning Systems, 31*(7), 2267–2279.
- Iyer, R., Li, Y., Li, H., Lewis, M., Sundar, R., & Sycara, K. (2018). Transparency and explanation in deep reinforcement learning neural networks. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 144–150.
- Jacovi, A., & Goldberg, Y. (2020). Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness?. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4198–4205.
- Jain, A., Koppula, H. S., Raghavan, B., Soh, S., & Saxena, A. (2015). Car that knows before you do: Anticipating maneuvers via learning temporal driving models. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3182–3190.
- Jain, S., & Wallace, B. C. (2019). Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3543–3556.
- Jesus, S., Belém, C., Balayan, V., Bento, J., Saleiro, P., Bizarro, P., & Gama, J. (2021). How can I choose an explainer? An application-grounded evaluation of post-hoc explanations. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, p. 805–815.
- Jha, S., Raman, V., Pinto, A., Sahai, T., & Francis, M. (2017). On learning sparse Boolean formulae for explaining AI decisions. In *NASA Formal Methods Symposium*, pp. 99–114.
- Jiang, H., Kim, B., Guan, M., & Gupta, M. (2018). To trust or not to trust a classifier. In *Advances in Neural Information Processing Systems*, pp. 5541–5552.
- Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., & Girshick, R. (2017). CLEVR: A diagnostic dataset for compositional language and elementary

- visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2901–2910.
- Kamiran, F., & Calders, T. (2010). Classification with no discrimination by preferential sampling. In *Proc. 19th Machine Learning Conf. Belgium and The Netherlands*, pp. 1–6.
- Kamiran, F., & Calders, T. (2012). Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, 33(1), 1–33.
- Kamishima, T., Akaho, S., & Sakuma, J. (2011). Fairness-aware learning through regularization approach. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pp. 643–650.
- Kang, D., Raghavan, D., Bailis, P., & Zaharia, M. (2018). Model assertions for debugging machine learning. In *NeurIPS MLSys Workshop*.
- Kearns, M., Neel, S., Roth, A., & Wu, Z. S. (2018). Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. In *International Conference on Machine Learning*, pp. 2569–2577.
- Kim, B., Rudin, C., & Shah, J. A. (2014). The Bayesian case model: A generative approach for case-based reasoning and prototype classification. In *Advances in Neural Information Processing Systems*, pp. 1952–1960.
- Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., et al. (2018a). Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In *International Conference on Machine Learning*, pp. 2673–2682.
- Kim, J., Rohrbach, A., Darrell, T., Canny, J., & Akata, Z. (2018b). Textual explanations for self-driving vehicles. In *Proceedings of the European Conference on Computer Vision*, pp. 563–578.
- Kindermans, P.-J., Hooker, S., Adebayo, J., Alber, M., Schütt, K. T., Dähne, S., Erhan, D., & Kim, B. (2019). The (un)reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pp. 267–280. Springer.
- Kindermans, P.-J., Schütt, K., Müller, K.-R., & Dähne, S. (2016). Investigating the influence of noise and distractors on the interpretation of neural networks. *ArXiv*, *abs/1611.07270*.
- Kindermans, P.-J., Schütt, K., Alber, M., Müller, K.-R., Erhan, D., Kim, B., & Dähne, S. (2018). Learning how to explain neural networks: PatternNet and PatternAttribution. In *International Conference on Learning Representations*.
- Koh, P. W., & Liang, P. (2017). Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1885–1894.
- Kolodner, J. L. (1992). An introduction to case-based reasoning. *Artificial Intelligence Review*, 6(1), 3–34.
- Krakovna, V., & Doshi-Velez, F. (2016). Increasing the interpretability of recurrent neural networks using hidden Markov models. *ArXiv*, *abs/1606.05320*.

- Kurakin, A., Goodfellow, I., & Bengio, S. (2016). Adversarial examples in the physical world. In *International Conference on Learning Representations*.
- Lage, I., Chen, E., He, J., Narayanan, M., Kim, B., Gershman, S., & Doshi-Velez, F. (2019). An evaluation of the human-interpretability of explanation. *ArXiv, abs/1902.00006*.
- Lapuschkin, S., Binder, A., Montavon, G., Müller, K.-R., & Samek, W. (2016). Analyzing classifiers: Fisher vectors and deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2912–2920.
- Lei, T., Barzilay, R., & Jaakkola, T. (2016). Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 107–117.
- Letarte, G., Paradis, F., Giguère, P., & Laviolette, F. (2018). Importance of self-attention for sentiment analysis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 267–275.
- Li, J., Monroe, W., & Jurafsky, D. (2016). Understanding neural networks through representation erasure. *ArXiv, abs/1612.08220*.
- Li, O., Liu, H., Chen, C., & Rudin, C. (2018a). Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
- Li, Y., Min, M. R., Yu, W., Hsieh, C.-J., Lee, T., & Kruus, E. (2018b). Optimal transport classifier: Defending against adversarial attacks by regularized deep embedding. *ArXiv, abs/1811.07950*.
- Lin, M., Chen, Q., & Yan, S. (2013). Network in network. In *International Conference on Learning Representations*.
- Lipton, Z. C. (2017). The doctor just won't accept that!. *ArXiv, abs/1711.08037*.
- Lipton, Z. C. (2018). The mythos of model interpretability. *Communications of the ACM*, 61(10), 36–43.
- Liu, H., Yin, Q., & Wang, W. Y. (2019). Towards explainable NLP: A generative explanation framework for text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5570–5581.
- Liu, S., Wang, X., Liu, M., & Zhu, J. (2017). Towards better analysis of machine learning models: A visual analytics perspective. *Visual Informatics*, 1(1), 48–56.
- Liu, X., Wang, X., & Matwin, S. (2018a). Improving the interpretability of deep neural networks with knowledge distillation. In *2018 IEEE International Conference on Data Mining Workshops*, pp. 905–912.
- Liu, X., Li, Y., Wu, C., & Hsieh, C.-J. (2018b). Adv-BNN: Improved adversarial defense through robust Bayesian neural network. In *International Conference on Learning Representations*.
- Louizos, C., Swersky, K., Li, Y., Welling, M., & Zemel, R. (2015). The variational fair autoencoder. In *International Conference on Learning Representations*.

- Lundberg, S. M., Erion, G. G., & Lee, S.-I. (2018). Consistent individualized feature attribution for tree ensembles. *ArXiv, abs/1802.03888*.
- Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pp. 4765–4774.
- Lundén, J., & Koivunen, V. (2016). Deep learning for HRRP-based target recognition in multistatic radar systems. In *IEEE Radar Conference*, pp. 1–6.
- Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421.
- Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.
- Marchette, C. E. P. D. J., & Socolinsky, J. G. D. A. (2003). Classification using class cover catch digraphs. *Journal of Classification, 20*, 3–23.
- Mascharka, D., Tran, P., Soklaski, R., & Majumdar, A. (2018). Transparency by design: Closing the gap between performance and interpretability in visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4942–4950.
- Masi, I., Wu, Y., Hassner, T., & Natarajan, P. (2018). Deep face recognition: A survey. In *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images*, pp. 471–478.
- Melis, D. A., & Jaakkola, T. (2018). Towards robust interpretability with self-explaining neural networks. In *Advances in Neural Information Processing Systems*, pp. 7775–7784.
- Meng, D., & Chen, H. (2017). MagNet: A two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 135–147.
- Menon, A. K., & Williamson, R. C. (2018). The cost of fairness in binary classification. In *Conference on Fairness, Accountability and Transparency*, pp. 107–118.
- Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence, 267*, 1–38.
- Miller, T. (2021). Contrastive explanation: a structural-model approach. *The Knowledge Engineering Review, 36*.
- Molnar, C. (2020). *Interpretable Machine Learning*. Lulu.com.
- Montavon, G., Lapuschkin, S., Binder, A., Samek, W., & Müller, K.-R. (2017). Explaining nonlinear classification decisions with deep Taylor decomposition. *Pattern Recognition, 65*, 211–222.
- Montavon, G., Samek, W., & Müller, K.-R. (2018). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing, 73*, 1–15.
- Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., & Frossard, P. (2017). Universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1765–1773.

- Moosavi-Dezfooli, S.-M., Fawzi, A., & Frossard, P. (2016). DeepFool: A simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582.
- Mueller, S. T., Hoffman, R. R., Clancey, W., Emrey, A., & Klein, G. (2019). Explanation in human-AI systems: A literature meta-review, synopsis of key ideas and publications, and bibliography for explainable AI. *ArXiv, abs/1902.01876*.
- Murdoch, W. J., Liu, P. J., & Yu, B. (2018). Beyond word importance: Contextual decomposition to extract interactions from LSTMs. In *International Conference on Learning Representations*.
- Murdoch, W. J., & Szlam, A. (2017). Automatic rule extraction from long short term memory networks. In *International Conference on Learning Representations*.
- Nemati, S., Holder, A., Razmi, F., Stanley, M. D., Clifford, G. D., & Buchman, T. G. (2018). An interpretable machine learning model for accurate prediction of sepsis in the ICU. *Critical Care Medicine, 46*(4), 547–553.
- Nguyen, A., Yosinski, J., & Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 427–436.
- Nie, L., Wang, M., Zhang, L., Yan, S., Zhang, B., & Chua, T.-S. (2015). Disease inference from health-related questions via sparse deep learning. *IEEE Transactions on Knowledge and Data Engineering, 27*(8), 2107–2119.
- Nie, W., Zhang, Y., & Patel, A. (2018). A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. In *International Conference on Machine Learning*, pp. 3806–3815.
- Oana-Maria, C., Brendan, S., Pasquale, M., Thomas, L., & Phil, B. (2020). Make up your mind! Adversarial generation of inconsistent natural language explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4157–4165.
- Olah, C., Mordvintsev, A., & Schubert, L. (2017). Feature visualization. *Distill, 2*(11).
- Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K., & Mordvintsev, A. (2018). The building blocks of interpretability. *Distill, 3*(3).
- Olfat, M., & Aswani, A. (2018). Spectral algorithms for computing fair support vector machines. In *International Conference on Artificial Intelligence and Statistics*, pp. 1933–1942.
- Ozbulak, U. (2019). PyTorch CNN visualizations. <https://github.com/utkuozbulak/pytorch-cnn-visualizations>.
- Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., & Swami, A. (2017). Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 506–519.
- Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., & Swami, A. (2016a). The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy*, pp. 372–387.

- Papernot, N., McDaniel, P., Wu, X., Jha, S., & Swami, A. (2016b). Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy*, pp. 582–597.
- Park, D. H., Hendricks, L. A., Akata, Z., Rohrbach, A., Schiele, B., Darrell, T., & Rohrbach, M. (2018). Multimodal explanations: Justifying decisions and pointing to the evidence. In *31st IEEE Conference on Computer Vision and Pattern Recognition*.
- Park, D. H., Hendricks, L. A., Akata, Z., Schiele, B., Darrell, T., & Rohrbach, M. (2016). Attentive explanations: Justifying decisions and pointing to the evidence. *ArXiv*, *abs/1612.04757*.
- Pérez-Suay, A., Laparra, V., Mateo-García, G., Muñoz-Marí, J., Gómez-Chova, L., & Camps-Valls, G. (2017). Fair kernel learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 339–355.
- Petsiuk, V., Das, A., & Saenko, K. (2018). RISE: Randomized input sampling for explanation of black-box models. In *British Machine Vision Conference*, p. 151.
- Pham, T. T., & Shen, Y. (2017). A deep causal inference approach to measuring the effects of forming group loans in online non-profit microfinance platform. *ArXiv*, *abs/1706.02795*.
- Pleiss, G., Raghavan, M., Wu, F., Kleinberg, J., & Weinberger, K. Q. (2017). On fairness and calibration. In *Advances in Neural Information Processing Systems*, pp. 5680–5689.
- Prasad, G., Nie, Y., Bansal, M., Jia, R., Kiela, D., & Williams, A. (2020). To what extent do human explanations of model behavior align with actual model behavior?. *ArXiv*, *abs/2012.13354*.
- Raghu, M., Gilmer, J., Yosinski, J., & Sohl-Dickstein, J. (2017). SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Advances in Neural Information Processing Systems*, pp. 6076–6085.
- Rajpurkar, P., Irvin, J., Ball, R. L., Zhu, K., Yang, B., Mehta, H., Duan, T., Ding, D., Bagul, A., Langlotz, C. P., et al. (2018). Deep learning for chest radiograph diagnosis: A retrospective comparison of the CheXNeXt algorithm to practicing radiologists. *PLOS Medicine*, *15*.
- Ras, G., van Gerven, M., & Haselager, P. (2018). Explanation methods in deep learning: Users, values, concerns and challenges. In *Explainable and Interpretable Models in Computer Vision and Machine Learning*, pp. 19–36. Springer.
- Ray, A., Yao, Y., Kumar, R., Divakaran, A., & Burachas, G. (2019). Can you explain that? Lucid explanations help human-AI collaborative image retrieval. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, Vol. 7, pp. 153–161.
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788.
- Ribeiro, M. T., Singh, S., & Guestrin, C. (2016a). Model-agnostic interpretability of machine learning. *ArXiv*, *abs/1606.05386*.

- Ribeiro, M. T., Singh, S., & Guestrin, C. (2016b). Nothing else matters: Model-agnostic explanations by identifying prediction invariance. *ArXiv, abs/1611.05817*.
- Ribeiro, M. T., Singh, S., & Guestrin, C. (2016c). 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*, pp. 1135–1144.
- Ribeiro, M. T., Singh, S., & Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
- Robnik-Šikonja, M., & Bohanec, M. (2018). Perturbation-based explanations of prediction models. In *Human and Machine Learning*, pp. 159–175. Springer.
- Robnik-Šikonja, M., & Kononenko, I. (2008). Explaining classifications for individual instances. *IEEE Transactions on Knowledge and Data Engineering, 20*(5), 589–600.
- Rolnick, D., Donti, P. L., Kaack, L. H., Kochanski, K., Lacoste, A., Sankaran, K., Ross, A. S., Milojevic-Dupont, N., Jaques, N., Waldman-Brown, A., et al. (2019). Tackling climate change with machine learning. *ArXiv, abs/1906.05433*.
- Rozsa, A., Rudd, E. M., & Boult, T. E. (2016). Adversarial diversity and hard positive generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 25–32.
- Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence, 1*(5), 206.
- Sabour, S., Cao, Y., Faghri, F., & Fleet, D. J. (2015). Adversarial manipulation of deep representations. In *International Conference on Learning Representations*.
- Saha, A., Subramanya, A., & Pirsavash, H. (2020). Hidden trigger backdoor attacks. *Proceedings of the AAAI Conference on Artificial Intelligence, 34*, 11957–11965.
- Samangouei, P., Kabkab, M., & Chellappa, R. (2018). Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*.
- Samek, W., Binder, A., Montavon, G., Lapuschkin, S., & Müller, K.-R. (2016). Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems, 28*(11), 2660–2673.
- Samek, W., Wiegand, T., & Müller, K.-R. (2017). Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *ArXiv, abs/1708.08296*.
- Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 618–626.
- Serrano, S., & Smith, N. A. (2019). Is attention interpretable?. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2931–2951.
- Shapley, L. (1953). A value for n-person games. *Contributions to the Theory of Games, 2.28*, 307–317.

- Shrikumar, A., Greenside, P., & Kundaje, A. (2017). Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 3145–3153.
- Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. In *International Conference on Learning Representations*.
- Sirignano, J., Sadhwani, A., & Giesecke, K. (2016). Deep learning for mortgage risk. *ArXiv*, *abs/1607.02470*.
- Sobol, I. M. (2001). Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and Computers in Simulation*, 55(1-3), 271–280.
- Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. In *International Conference on Learning Representations*.
- Su, J., Vargas, D. V., & Sakurai, K. (2019). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5), 828–841.
- Sundararajan, M., Taly, A., & Yan, Q. (2016). Gradients of counterfactuals. *ArXiv*, *abs/1611.02639*.
- Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 3319–3328.
- Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. In *International Conference on Learning Representations*.
- Tabacof, P., & Valle, E. (2016). Exploring the space of adversarial images. In *2016 International Joint Conference on Neural Networks*, pp. 426–433.
- Tan, S., Caruana, R., Hooker, G., Koch, P., & Gordo, A. (2018). Learning global additive explanations for neural nets using model distillation. *ArXiv*, *abs/1801.08640*.
- Teney, D., Anderson, P., He, X., & van den Hengel, A. (2018). Tips and tricks for visual question answering: Learnings from the 2017 challenge. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4223–4232.
- Tjoa, E., & Guan, C. (2020). A survey on explainable artificial intelligence (XAI): Toward medical XAI. In *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21.
- Tulshan, A. S., & Dhage, S. N. (2018). Survey on virtual assistant: Google Assistant, Siri, Cortana, Alexa. In *International Symposium on Signal Processing and Intelligent Recognition Systems*, pp. 190–201.
- Vashishth, S., Upadhyay, S., Tomar, G. S., & Faruqui, M. (2019). Attention interpretability across NLP tasks. *ArXiv*, *abs/1909.11218*.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008.

- Verma, S., Dickerson, J., & Hines, K. (2020). Counterfactual Explanations for Machine Learning: A Review. *ArXiv, abs/2010.10596*.
- Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3156–3164.
- Vu, M. N., Nguyen, T. D., Phan, N., Gera, R., & Thai, M. T. (2019). c-Eval: A unified metric to evaluate feature-based explanations via perturbation. *ArXiv, abs/1906.02032*.
- Wang, Y., Huang, M., Zhao, L., et al. (2016). Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 606–615.
- Wiegreffe, S., & Pinter, Y. (2019). Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 11–20.
- Wolf, L., Galanti, T., & Hazan, T. (2019). A formal approach to explainability. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 255–261.
- Woodworth, B., Gunasekar, S., Ohannessian, M. I., & Srebro, N. (2017). Learning non-discriminatory predictors. *ArXiv, abs/1702.06081*.
- Xie, C., Wang, J., Zhang, Z., Ren, Z., & Yuille, A. (2017). Mitigating adversarial effects through randomization. In *International Conference on Learning Representations*.
- Xie, N., Lai, F., Doran, D., & Kadav, A. (2019). Visual entailment: A novel task for fine-grained image understanding. *ArXiv, abs/1901.06706*.
- Xie, N., Sarker, M. K., Doran, D., Hitzler, P., & Raymer, M. (2017). Relating input concepts to convolutional neural network decisions. *ArXiv, abs/1711.08006*.
- Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., & Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pp. 2048–2057.
- Yim, J., Joo, D., Bae, J., & Kim, J. (2017). A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *The IEEE Conference on Computer Vision and Pattern Recognition*.
- Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., & Lipson, H. (2015). Understanding neural networks through deep visualization. *ArXiv, abs/1506.06579*.
- Yuan, X., He, P., Zhu, Q., & Li, X. (2019). Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems, 30*(9), 2805–2824.
- Zafar, M. B., Valera, I., Gomez Rodriguez, M., & Gummadi, K. P. (2017a). Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *Proceedings of the 26th International Conference on World Wide Web*, pp. 1171–1180.
- Zafar, M. B., Valera, I., Rodriguez, M., Gummadi, K., & Weller, A. (2017b). From parity to preference-based notions of fairness in classification. In *Advances in Neural Information Processing Systems*, pp. 229–239.

- Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pp. 818–833.
- Zeiler, M. D., Taylor, G. W., & Fergus, R. (2011). Adaptive deconvolutional networks for mid and high level feature learning. In *2011 International Conference on Computer Vision*, pp. 2018–2025.
- Zellers, R., Bisk, Y., Farhadi, A., & Choi, Y. (2019). From recognition to cognition: Visual commonsense reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6720–6731.
- Zemel, R., Wu, Y., Swersky, K., Pitassi, T., & Dwork, C. (2013). Learning fair representations. In *International Conference on Machine Learning*, pp. 325–333.
- Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*.
- Zhang, Q.-s., & Zhu, S.-C. (2018). Visual interpretability for deep learning: A survey. *Frontiers of Information Technology & Electronic Engineering*, 19(1), 27–39.
- Zhang, Q., Cao, R., Shi, F., Wu, Y. N., & Zhu, S.-C. (2018). Interpreting CNN knowledge via an explanatory graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
- Zhang, Q., Cao, R., Wu, Y. N., & Zhu, S.-C. (2017). Growing interpretable part graphs on ConvNets via multi-shot learning. In *Proceedings of AAAI Conference on Artificial Intelligence*, pp. 2898–2906.
- Zhang, Q., Yang, Y., Ma, H., & Wu, Y. N. (2019a). Interpreting CNNs via decision trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6261–6270.
- Zhang, W. E., Sheng, Q. Z., & Alhazmi, A. A. F. (2019b). Generating textual adversarial examples for deep learning models: A survey. *ArXiv*, abs/1901.06796.
- Zhao, Z., Dua, D., & Singh, S. (2017). Generating natural adversarial examples. In *International Conference on Learning Representations*.
- Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2014). Object detectors emerge in deep scene CNNs. In *International Conference on Learning Representations*.
- Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2921–2929.
- Zintgraf, L. M., Cohen, T. S., Adel, T., & Welling, M. (2017). Visualizing deep neural network decisions: Prediction difference analysis. In *International Conference on Learning Representations*.