Philosophy & Technology (2021) 34:265–288
https://doi.org/10.1007/s13347-019-00382-7

RESEARCH ARTICLE

Solving the Black Box Problem: A Normative Framework for Explainable Artificial Intelligence

Carlos Zednik[1]

Received: 11 March 2019 / Accepted: 14 October 2019 / Published online: 20 December 2019

Abstract
Many of the computing systems programmed using Machine Learning are *opaque*: it is difficult to know why they do what they do or how they work. *Explainable Artificial Intelligence* aims to develop analytic techniques that render opaque computing systems transparent, but lacks a normative framework with which to evaluate these techniques' explanatory successes. The aim of the present discussion is to develop such a framework, paying particular attention to different stakeholders' distinct explanatory requirements. Building on an analysis of "opacity" from philosophy of science, this framework is modeled after accounts of explanation in cognitive science. The framework distinguishes between the explanation-seeking questions that are likely to be asked by different stakeholders, and specifies the general ways in which these questions should be answered so as to allow these stakeholders to perform their roles in the *Machine Learning ecosystem*. By applying the normative framework to recently developed techniques such as *input heatmapping*, *feature-detector visualization*, and *diagnostic classification*, it is possible to determine whether and to what extent techniques from Explainable Artificial Intelligence can be used to render opaque computing systems transparent and, thus, whether they can be used to solve the *Black Box Problem*.

Keywords Artificial intelligence · Black box problem · Epistemic opacity · Explainable artificial intelligence · Levels of analysis · Machine learning · Scientific explanation

1 Introduction

Computing systems programmed using Machine Learning (ML) are increasingly capable of solving complex problems in Artificial Intelligence (AI). Unfortunately, these systems remain characteristically *opaque*: it is difficult to "look inside" so as to explain their behavior.

✉ Carlos Zednik
carlos.zednik@ovgu.de; http://carloszednik.net/

[1] Otto-von-Guericke-Universität Magdeburg, Magdeburg, Germany

Opacity is the heart of the *Black Box Problem in AI*—a problem with practical, legal, and theoretical consequences. Practically, end users are less likely to trust and cede control to machines whose workings they do not understand (Burrell 2016; Ribeiro et al. 2016), and software developers may be unable to intervene in order to quickly and systematically improve performance (Hohman et al. 2018). Legally, opacity prevents regulatory bodies from determining whether a particular system processes data fairly and securely (Rieder and Simon 2017) and may hinder end users from exercising their rights under the European Union's *General Data Protection Regulation* (Goodman and Flaxman 2016; but cf. Wachter et al. 2017). Theoretically, the Black Box Problem makes it difficult to evaluate the potential similarity between artificial neural networks and biological brains (Buckner 2018), as well as to determine the extent to which computers being developed using Machine Learning should be considered genuinely intelligent (Zednik 2018).

Investigators within the *Explainable Artificial Intelligence* (*Explainable AI* or XAI) research program intend to ward off these consequences through the use of analytic techniques capable of rendering opaque computing systems *transparent*.[1] Although this research program has already commanded significant attention (see, e.g., Burrell 2016; Doran et al. 2017; Lipton 2016; Ras et al. 2018; Zerilli et al. 2018), conceptual questions remain about the dual notions of "opacity" and "transparency" and, thus, about what it actually means to *explain* a computing system's behavior. Although the Black Box Problem may be rooted in the difficulties software developers face when attempting to explain a system's behavior by acquiring knowledge of its learnable parameters, other stakeholders are equally likely to seek explanations, and other kinds of knowledge may equally serve explanatory purposes. For example, decision subjects are most likely to benefit from explanations that cite the reasons for a system's output, while regulatory bodies may be best served by understanding the environmental regularities that are being tracked. Operators may only require explanations that clarify the system's input and output states, while hardware engineers may require explanations that characterize the physical structures in which computational operations are carried out. In general, although "opacity," "transparency," and "explanation" mean different things to different stakeholders, there is as of yet no general account of what these terms actually mean for different stakeholders in the *ML ecosystem* (Tomsett et al. 2018).

The present discussion analyzes "opacity" as a particular stakeholder's lack of a certain kind of knowledge. Accordingly, "transparency" is the possession of such knowledge, and "explanation" is the process by which a stakeholder acquires the relevant kind of knowledge. This analysis provides the conceptual foundation for a *normative framework* for Explainable AI: A framework that specifies the kind(s) of knowledge that *should* be provided to render opaque computing systems transparent for particular stakeholders. Such a framework is needed to determine whether, to what extent, and for whom the the behavior of opaque computing systems can eventually be explained, and thus, whether the Black Box Problem can eventually be solved.

[1] It is worth distinguishing two distinct streams within the Explainable AI research program. The present discussion focuses on attempts to *solve* the Black Box Problem by analyzing computing systems so as to render them transparent post hoc, i.e., after they have been developed. In contrast, the discussion will not consider efforts to *avoid* the Black Box Problem altogether, by modifying the relevant ML methods so that the computers being programmed do not become opaque in the first place (for discussion see, e.g., Doran et al. 2017).

The discussion proceeds as follows. Section 2 builds on previous philosophical work by Paul Humphreys (2009) to analyze the oft-used but still poorly understood notions of "opacity" and "transparency." Section 3 then introduces Tomsett et al.'s (2018) notion of an "ML ecosystem" to distinguish between the stakeholders who are most likely to be affected by a system's opacity, and thus, are most likely to seek explanations of its behavior. It also introduces David Marr's (1982) *levels of analysis* account of explanation in cognitive science so as to better understand these stakeholders' distinct explanatory requirements. Indeed, different stakeholders in the ML ecosystem can be aligned with different questions in Marr's account—questions about *what*, *why*, *how*, and *where* a computer program is carried out. Sections 4 and 5 then introduce and evaluate the explanatory successes of several analytic techniques from Explainable AI, with a particular focus on *layer-wise relevance propagation* (Montavon et al. 2018), *feature-detector visualization* (Bau et al. 2018), and *diagnostic classification* (Hupkes et al. 2018).[2] Indeed, these sections show that different XAI techniques are capable of answering distinct explanation-seeking questions and, thus, are likely to satisfy different stakeholders' explanatory requirements.

2 The Black Box Problem in Artificial Intelligence

2.1 From Machine Learning to the Black Box Problem

The Black Box Problem arises when the computing systems being developed in AI are *opaque*. This metaphorical way of speaking is grounded in the intuition that a system's behavior can be explained by "looking inside." Of course, most computing systems are constructed from well-understood hardware components that afford no literal obstacle to looking inside. Nevertheless, they might be considered opaque in the sense that it is difficult to know exactly how these systems are *programmed*.

In order to go beyond the metaphor, it is important to recognize that Machine Learning is just one approach among many for programming computers that solve complex problems in AI (Russell, Norvig, & Davis, 2010). In most traditional approaches, software developers write the source code needed for a computer to solve a specific AI problem. In contrast, developers in ML program computers to find solutions on their own. To this end, ML developers specify basic architectural principles such as whether a system takes the form of a deep neural network, a support vector machine, or a decision tree, among others. Moreover, they specify an appropriate learning algorithm and select a suitable learning environment in which to attribute values to the system's *learnable parameters* (e.g., the connection weights within an artificial neural network). However, ML developers do not set these values themselves. Instead, the parameter values are set by the learning algorithm, through its engagement with the learning environment. Thus, in contrast to their colleagues working in other approaches in AI, ML developers do not actually specify how an AI problem is solved but merely specify the conditions under which a solution may eventually be found.

[2] This list is by no means comprehensive. Indeed, XAI is an incredibly dynamic field in which original and increasingly powerful techniques are being developed almost daily. Although not all relevant XAI techniques can be considered, the normative framework being developed here is meant to apply generally, including to unconsidered techniques and to techniques that do not yet exist.

This relative lack of influence on the way in which problems are actually solved is one of the principal advantages of Machine Learning over traditional approaches in AI. Because of it, the solutions that are actually identified may be highly unintuitive and subtle (as well as effective) and unlikely to be found using more traditional methods.[3] But this relative lack of influence also begets problems. Because ML developers do not themselves set a system's parameter values, they might not actually know what these values are. But even if they do know the values of individual parameters, the fact that these values are often numerous and nonlinearly coupled means that developers cannot readily track the way individual inputs are transformed into specific outputs. For this reason, they are often unable to understand the solution and unable to intervene so as to systematically and predictably change the system's behavior.[4]

Although software developers' relative lack of influence on the way in which an AI problem is solved may be the root cause of opacity in Machine Learning, "opacity" cannot be defined in terms of a developer's (lack of) knowledge of learnable parameters. In many cases, rendering an opaque system transparent will not actually involve acquiring knowledge of parameter values at all but will instead require knowledge of the environmental patterns and regularities that are being tracked and of the abstract representational structures that are tracking them (Buckner 2018). Moreover, in many cases, stakeholders other than developers will require explanations of a particular system's behavior, despite being unwilling or unable to acquire knowledge of learnable parameters (Burrell 2016). Thus, "opacity" is a nuanced concept that deserves careful analysis.

2.2 Opacity, Transparency, and Explanation

Perhaps the most influential analysis of "opacity" is due to Paul Humphreys (2009). Although Humphreys' work is primarily concerned with computer simulations in scientific disciplines such as high-energy physics and molecular biology, his analysis of "opacity" is also a useful starting point for discussions of the Black Box Problem in AI. On Humphreys' analysis, computing systems are:

"opaque relative to a cognitive agent *X* at time *t* just in case *X* does not know at *t* all of the epistemically relevant elements of the [system]." (Humphreys 2009, p. 618)[5]

[3] Their ability to produce such solutions is perhaps the main reason for using ML methods in many different problem domains. That said, it is important to recognize that ML methods are by no means all-conquering. Many important AI problems remain unsolved, and in many cases, it is unclear whether, and if so how, ML methods could ever be used to solve them. Indeed, in many problem domains, traditional AI methods remain far more effective (for discussion see, e.g., Lake et al. 2017; Marcus 2018)

[4] Indeed, although ML developers typically have access to a system's learnable parameter values, they tend to be the first to call for more illuminating explanations that allow them to, for example, demonstrate that the system does in fact do what it is supposed to do (see Sect. 4 below) or to improve its behavior when it does not (see Sect. 5 below; Hohman et al. 2018).

[5] Humphreys subsequently introduces the notion of "*essential* epistemic opacity," which applies to systems whose epistemically relevant elements are not only unknown to the agent but are in fact impossible to know by that agent (Humphreys 2009, p. 618). Notably, Humphreys' notion of possibility is not logical but practical—it depends on an agent's limited time, money, and computational resources, among other things (Durán and Formanek 2018). For this reason, analytic techniques from XAI that allow an agent to use these resources more efficiently are poised to broaden the scope of the possible all while allowing that some systems may remain opaque even after the most powerful techniques have been applied. That said, the present discussion need not speculate about the scope of the possible. Rather, its aim is to show how systems that *can* be rendered transparent *should* be rendered transparent.

Two features of this analysis are worth highlighting. First, opacity is *agent-relative*. A computing system is never opaque in and of itself but opaque only with respect to some particular agent. Second, opacity is an *epistemic* property: it concerns the (lack of) a certain kind of knowledge. According to Humphreys, this knowledge is knowledge of the system's *epistemically relevant elements* (EREs). Although Humphreys does not further analyze the notion of an "epistemically relevant element," it can be fleshed out in a way that suits current purposes. Thus, an *element* can be understood as, for example, a step in the process of transforming inputs to outputs, or as a momentary state transition within the system's overall evolution over time. An *epistemically relevant* element is one which is not only known to the agent but which can be cited by him or her to explain the occurrence of some other element or of the system's overall output.

Given that a system is opaque when an agent *lacks* knowledge of its EREs, it is transparent when the agent *possesses* such knowledge. Explanation, then, can be viewed as the *acquisition* of knowledge of a system's EREs. Note, however, that the term "explanation" is intentionally ambiguous. Any computing system's behavior can be explained in many different ways, depending on the particular kind of ERE that is being identified. In particular, electrochemical explanations might appeal to the changing magnetization of hardware registers, mathematical explanations might appeal to the manipulation of binary strings, and rational explanations might appeal to the system's goals and representational states. Notably, in affording such a diversity of explanations and EREs, computing systems resemble humans and other biological cognizers. Indeed, like the computing systems being developed using Machine Learning, biological cognizers can be explained electrochemically (e.g., by reference to neurobiological mechanisms), mathematically (e.g., by reference to computational processes), and rationally (e.g., by reference to beliefs and desires, see, e.g., Dennett, 1987). Staunchly reductionist or eliminativist philosophical tendencies notwithstanding (e.g., Bickle 2006; Churchland 1981), all of these explanations are equally legitimate, and all of the EREs invoked in these explanations can be considered equally real.

But although many different explanations are equally legitimate, they are unequal in other respects. Most notably, different explanations are likely to serve different purposes and, for this reason, are likely to be appropriate for different stakeholders. Indeed, depending on what a particular stakeholder is tasked with doing, they are likely to require a different kind of knowledge to do it and, thus, to seek a different kind of explanation. But who are these stakeholders, and what exactly are they tasked with doing?

3 From Machine Learning to Marr

3.1 Who Are the Stakeholders, and What Are they Tasked with Doing?

Tomsett et al. (2018) provide a helpful taxonomy of stakeholders in the *ML ecosystem*, that is, stakeholders who depend on or interact with a computing system programmed using Machine Learning. Six kinds of stakeholders are distinguished according to their designated roles within the ecosystem, and Tomsett et al. illustrate these roles by invoking as an example a *loan risk assessment system*. Indeed, the loan risk assessment system is a typical, albeit relatively simple, ML application: A supervised learning algorithm can be used quite straightforwardly to correlate previous applicants' personal data such as income, age, and home address with their eventual ability to repay loans in a timely manner. On the basis of these correlations, the system can take a new individual's personal data as input and generate an output to estimate the financial risk a bank would incur by accepting that individual's application.

In the risk assessor's ecosystem, bank employees are *operators* and/or *executors*. Operators are agents who "provide the system with inputs, and directly receive the system's outputs" (Tomsett et al. 2018, p. 10). These are likely to include bank tellers tasked with entering a particular applicant's personal data into the system and with receiving the system's output. In contrast, executors are "agents who make decisions that are informed by the machine learning system" (Tomsett et al. 2018, p. 10). These are likely to include back-office employees who rely on the system's output to make data-driven decisions about whether or not to accept a particular application.

Another important agent in the ML ecosystem is the *decision subject*. In the present example, the decision subject is the loan applicant: the individual whose data are being processed for the purposes of assessing risk and who is subject to the executor's final decision. Decision subjects are distinguished from *data subjects*, individuals whose personal data are contained in the learning environment. In the loan application example, data subjects include previous applicants whose personal data and loan repayment behavior ground the risk assessor's learned correlations.

Creators are the developers of the computing system. These include the software developers responsible for designing the learning algorithm and for selecting the learning environment. They are also likely to include system administrators responsible for maintaining and possibly fine-tuning the system once it has been deployed. Of course, creators might also include hardware engineers tasked with building and maintaining the hardware in which the risk assessor is implemented. That said, many current ML applications are driven by standard-issue hardware components whose workings require no ML-specific knowledge.[6] Perhaps for this reason, Tomsett et al. (2018) do not explicitly consider creators of this kind.

Finally, *examiners* are "agents tasked with compliance/safety-testing, auditing, or forensically investigating a system" (Tomsett et al. 2018, p. 13). Thus, examiners are likely to include regulatory bodies charged with determining that the risk assessment system conforms to, for example, data privacy regulations, anti-discrimination laws, and policies designed to prevent financial fraud. Given legislative developments such as the EU *General Data Protection Regulation* (GDPR, European Commission 2016), examiners are likely to play an increasingly prominent role in future ML ecosystems.

[6] Applications that depend on ML-specific hardware components are considered in Sect. 5.

3.2 What Are the Epistemically Relevant Elements?

The Black Box Problem arises when a computing system's opacity—lack of knowledge of certain EREs—prevents stakeholders from performing their designated roles within the ML ecosystem. Section 2 already suggested that the knowledge required differs between stakeholders. This suggestion can now be confirmed in the context of the loan risk assessment system. Bank tellers tasked with operating the system will not be able to perform this role if they do not know that the DOB input variable needs to be formatted YYYYMMDD rather than DDMMYYYY and that the relevant output variable (i.e., the one to be communicated to the back-office examiner) is RISK. ML developers charged with engineering, maintaining, and improving the risk assessor's behavior will be unable to do so if they do not know the variables—learnable parameters and/or abstract representational structures—that mediate the transformation of inputs to outputs. Most other agents are less likely to require knowledge of the system's variables than knowledge of the environmental features that these variables represent. In particular, executors, examiners, data subjects, and decision subjects may not need to know much about variables such as INCOME, DOB, HOME_ADDRESS, and RISK but will have to know something about (their own or their customers') income levels and demographic information and, eventually, about the level of financial risk associated therewith.

As stated previously, these differences in the knowledge required by different stakeholders translates directly to a difference in explanatory requirements: in order to perform their designated roles, different stakeholders require explanations that center on different kinds of epistemically relevant elements.[7] But although these differences may be apparent within the context of the simple loan application example, a more general framework is required to specify these differences in a more systematic way.

3.3 Toward a Normative Framework for Explainable AI

In order to develop such a framework, it is instructive to look at cognitive science. Despite obvious differences to the computing systems being programmed in Machine Learning, biological cognizers can equally be viewed as opaque "black boxes." Moreover, as has already been suggested above, biological cognizers and ML-programmed computers afford a similar diversity of explanations and epistemically relevant elements. Of course, these similarities should not come as a surprise: biological cognizers have long been viewed as computing systems in their own right (Pylyshyn 1984), and the problems of Artificial Intelligence are traditionally defined in terms of the capacities possessed by humans and other intelligent beings (Minsky, 1968).

Perhaps the most widely influential account of explanation in cognitive science is David Marr's (1982) *levels of analysis* framework. Despite its age, Marr's account

[7] Different explanatory requirements might also stem from other differences, such as differences in background knowledge, abilities, and skills. Although these additional differences are undeniably interesting and relevant, the present discussion will not consider them further and will instead only focus on the differences that arise from stakeholders' distinct roles in the ML ecosystem.

remains influential in cognitive science today and does so in part because of its multifaceted character. Specifically, Marr's account centers on the answering of different explanation-seeking *questions* at three distinct *levels of analysis.* Whereas the computational level addresses questions about *what* a system is doing and *why* (Shagrir 2010), the algorithmic level concerns questions about *how* the system does what it does (McClamrock 1991), and the implementational level answers questions about *where* the relevant computational operations are realized (Zednik 2017). This multifaceted character recommends Marr's framework as a starting point from which to capture the differences between different ways of explaining the behavior of opaque computing systems. Indeed, insofar as different stakeholders can be thought to ask different explanation-seeking questions, the characteristic answers described by Marr might be used to identify the EREs that must be known in order to satisfy these stakeholders' distinct explanatory demands.

Using Marr's framework in this way is tantamount to specifying a normative framework for Explainable AI. Insofar as Explainable AI aims to develop analytic techniques for rendering opaque computing systems transparent, it is helpful to possess a set of articulated standards with which to determine whether this aim has actually been achieved. By identifying the EREs that must be known in order to satisfy different stakeholders' explanatory requirements, it is possible to evaluate individual techniques' explanatory successes. In particular, it is possible to determine whether these techniques do in fact deliver knowledge of the relevant kind of ERE. Insofar as this knowledge corresponds to the knowledge required to perform a specific role within in the ML ecosystem, the XAI technique in question satisfies the relevant stakeholder's explanatory requirements. For this stakeholder, the XAI technique can thus be said to render the computing system transparent and, indeed, to solve the Black Box Problem.

4 Description and Interpretation: The Computational Level

4.1 Questions About What and Why

Marr's *computational level of analysis* centers on questions about *what* a system is doing and on questions about *why* it does what it does. These questions are importantly different, though intimately related. Questions about *what* a system is doing call for a *description* of that system's overall behavior in terms of, for example, a transition function *f* in which "input" states are mapped onto corresponding "output" states (Fig. 1). In the context of the loan risk assessment system, what-questions are answered by specifying the value of the RISK variable that is generated for any particular combination of input variables such as INCOME, DOB, and HOME_ADDRESS.

In contrast, questions about *why* a system does what it does are questions about that behavior's "appropriateness" within some particular environment (Marr 1982, p. 24f). Thus, these questions call for an *interpretation* of the system's behavior in terms of recognizable features of the environment, be this the learning environment that (together with the learning algorithm) determines the system's behavior, or the behavioral environment in which the system is eventually deployed. Insofar as what-questions are answered by specifying a transition function *f* that maps "input" states onto corresponding "output" states, why-questions concern the environmental regularity or correlation *f'* that obtains between the inputs and outputs themselves.[8] Specifically, answering a why-question involves showing that there exists a correspondence between *f* and *f'* (Shagrir 2010. See also Fig. 1). Thus, to continue with the running example, in the loan application scenario, why-questions are not answered by describing the values of variables such as INCOME, DOB, HOME_ADDRESS, and RISK but rather by interpreting these values in terms of features such as a particular applicant's income, date of birth, home address, and level of financial risk.

It may be helpful to view the correspondence between systems and their environments semantically, that is, in representational terms. The system's "input" and "output" states may be thought to represent the input received from the environment, as well as the output being generated in response. Likewise, the state transition *f* might be thought to track the regularity or correlation *f'* that obtains between those inputs and outputs. Although more can and should be said about the correspondence that obtains between a computing system and its environment (for discussion see, e.g., Shagrir 2010), for current purposes, it suffices to say that whereas what-questions concern intrinsic properties of the computing system itself (often, its representational vehicles), why-questions concern features of the surrounding environment (the corresponding representational contents).

4.2 Operators, Executors, Examiners, Data and Decision Subjects

This admittedly brief presentation of the computational level can be used to align different stakeholders with specific explanation-seeking questions. For this reason, it can be used to

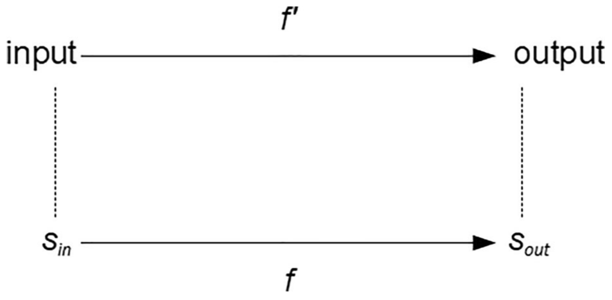
$$\begin{tikzcd}
\text{input} \arrow[r, "f'"] \arrow[d, dotted, no head] & \text{output} \arrow[d, dotted, no head] \\
s_{in} \arrow[r, "f"'] & s_{out}
\end{tikzcd}$$

Fig. 1 S_{in} and S_{out} are the input and output states of the computing system respectively;input and output are the corresponding features of the environment .The solid arrow at the bottom designates the overall behavior of the system presumably realized in some physical structure. The dotted arrow designates a representation (or other kind of correspondence) relationship between the system and its environment. What-questions concern the transition function *f* from s_{in} to s_{out}.Why-questions, in contrast, concern the relationship *f* between input and output. Adapted from Shagrir (2010).

[8] The present account is agnostic with respect to which *kind* of regularity or correlation actually obtains between inputs and outputs. In particular, whereas in some cases *f* might be a mere statistical correlation, in others *f* might pertain to a bona fide causal regularity. Although recent work has emphasized the advantages of ML-programmed systems that learn to track genuinely causal relationships (see, e.g., Wachter et al. 2018), much interesting work is also based on (possibly quite surprising) statistical correlations.

better understand the kinds of EREs these agents should come to know in order to perform their designated roles within the ML ecosystem.

Insofar as they are tasked with entering inputs and receiving outputs, operators are most likely to seek answers to questions about *what* a computing system is doing. That is, their explanatory requirements are satisfied—the system is rendered suitably transparent—by describing the inputs that must be entered and the outputs that are generated. In contrast, executors and examiners are far more likely to be concerned with why-questions, for which a system's behavior must be interpreted. Upon learning that the risk assessment system computes a RISK value of 0.794, a back-office executor's primary responsibility is to interpret that value as an indicator of significant financial risk. Similarly, it is an examiner's responsibility to determine whether a particular assessment has been generated legitimately or because the system discriminates by associating a foreign place of birth with a high level of financial risk.[9]

Data and decision subjects are similarly concerned with questions about *why* rather than with questions about *what*. Since coming into force in May 2018, the GDPR empowers data and decision subjects to seek answers to why-questions but does not similarly extend to what-questions. In particular, the GDPR's *right to information* allows data subjects to know which personal information is represented, but not how that information is entered, represented, and manipulated. Similarly, the GDPR's *right to explanation* might allow decision subjects to know which factors contribute to a particular outcome (e.g., a low income leads to a denied application), but not to the particular way in which those factors are represented, nor to the way in which the outcomes are actually calculated.[10] Notably, the rationale for focusing on questions about *why* rather than on questions about *what* (or on questions about *how*, see Sect. 5) is compelling: whereas my personal data belong to me, the data structures and processes that are used to represent and manipulate those data are the property of the AI service provider (see also Burrell 2016).

In summary, the distinction between what-questions and why-questions at the computational level captures an important distinction between two different ways of rendering a computing system transparent, each of which is appropriate for different stakeholders within the ML ecosystem. Whereas operators seek to render a system transparent by asking *what* it does and by describing its "input" and "output" states, several other agents do so by asking *why* it does what it does and interpreting those states in terms of environmental features and regularities. Perhaps surprisingly given the "black box" metaphor, this means that for several stakeholders in the ML

[9] Although executors and examiners are equally motivated by answering why-questions, they may be concerned with why-questions of different scope. Whereas executors must decide on an individual case ("the applicant's income is too low"), an examiner is more likely to be interested in a general tendency ("applicants with low income are rejected systematically"). Although both may be considered answers to why-questions insofar as the relevant EREs are features of the environment, the former is narrower in scope than the latter.

[10] It is a point of contention to what extent the GDPR's right to explanation constitutes a right at all and, if so, what it actually guarantees (Wachter et al. 2017). Although the former is a legal question that goes beyond the scope of the present discussion, the latter is a normative question that may benefit from the present discussion. In particular, Goodman and Flaxman (2016) argue that the GDPR grants data subjects (and decision subjects) the right to acquire "meaningful information about the logic involved." However, what exactly is meant by "logic" in this context remains unclear. The present discussion implies that the relevant stakeholders should be primarily concerned with why-questions and that in this sense, the "logic" should be specified in terms of learned regularities between environmental factors. Moreover, the present discussion suggests that AI service providers may, for legal reasons, be compelled to deploy XAI techniques capable of answering why-questions.

ecosystem, rendering a computing system transparent does not involve "looking inside" at all but rather "looking out" at the environment in which that system's behavior is learned and performed.

4.3 Input Heatmapping

To what extent can this analysis of what- and why-questions be used to evaluate the explanatory successes of current XAI techniques? Consider *input heatmapping*, a popular technique for highlighting features of a system's input that are particularly relevant to, or predictive of, its output. Some of the most compelling examples of this technique come from the domain of machine vision, in which deep neural networks are used to classify pixelated input images according to the people, objects, properties, or situations they depict.[11] There, input heatmapping typically involves the generation of visualizations—heatmaps—that emphasize the particular pixels or pixel regions that are most responsible for a particular classification (Fig. 2).

One concrete approach for developing heatmaps for artificial neural networks is *layer-wise relevance propagation* (LRP, Montavon et al. 2018). This method deploys a subroutine of the popular backpropagation learning algorithm, in which individual unit activations and connection weights are used to calculate the responsibility that the individual units of an "upstream" layer l_i bear for producing particular levels of activity in the subsequent "downstream" layer l_{i+1}. Given a particular classification at the network's output layer, this subroutine can be deployed layer-by-layer until responsibility values are calculated for every unit in the network's input layer. Insofar as these units correspond to, for example, pixels of a particular input image, these responsibility values can be used to generate a heatmap that highlights those pixels or pixel regions that bear the greatest responsibility for the final classification.

Input heatmaps provide particularly fine-grained answers to questions about *what* a computing system is doing. Recall that questions of this sort are answered by specifying the transition function f that obtains between a system's "input" and "output" states. Input heatmaps can help specify f, not in terms of the "input" state as a whole (e.g., a whole pixelated image) but in terms of a limited number of elements within that state (e.g., individual pixels). Such fine-grained answers to what-questions are particularly useful for operators. In particular, they greatly enhance an operator's ability to identify the most likely sources of error. If an error is detected in the "output" state (e.g., the RISK value is inappropriately high), it is more likely to result from high-responsibility elements of the "input" state (e.g., a wrongly formatted DOB value) than from low-responsibility elements (e.g., a misspelled surname). That said, such fine-grained answers to what-questions might also be exploited by malicious operators such as hackers. Indeed, input heatmaps can be used to design *adversarial inputs* that appear

[11] Although machine vision may be the domain in which input heatmaps are most intuitive, they may also be used in other domains. For example, input heatmaps may be constructed for audio inputs, highlighting the moments within an audio recording that are most responsible for classifying that recording according to musical genre. Moreover, although LRP is specifically designed to work with artificial neural networks (i.e., it is *model-specific*), other methods can be used to generate input heatmaps for other kinds of systems. Thus, input heatmapping in general can be viewed as a general-purpose (i.e., *model-agnostic*) XAI technique for answering what- and why-questions.

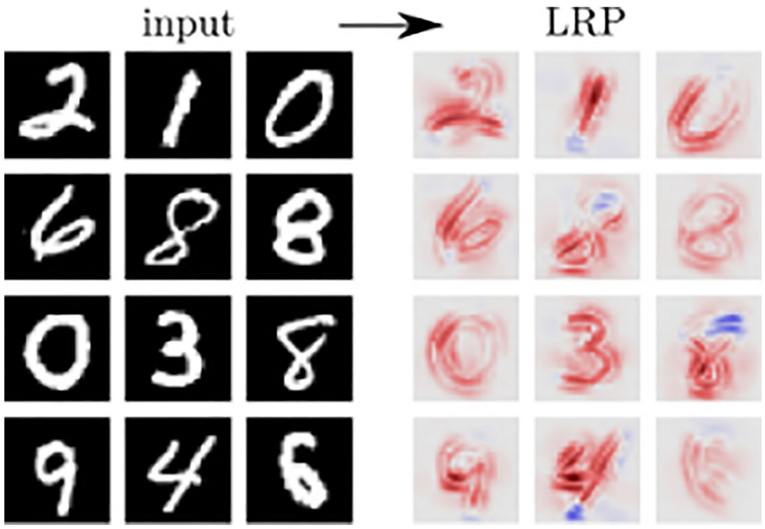

Fig. 2 Input heatmaps (right) for various input images (left), generated by applying the method of layer-wise relevance propagation (LRP) to a deep neural network capable of recognizing handwritten digits. Reproduced from Montavon et al. (2018).

typical to human observers, but that nevertheless produce radically divergent (and thus, potentially exploitable) outputs due to minor changes to some high-responsibility elements (Szegedy et al. 2013).

Notably, input heatmaps can also be used by creators. Creators are regularly tasked with improving or otherwise changing a system's overall behavior and, thus, with changing the system's transition function *f*. Input heatmap's fine-grained answers to what-questions can be used by creators to identify the specific modifications that must be made. Thus, for example, in order to maximize a system's processing efficiency and minimize its memory load, an input heatmap might be used to determine whether certain aspects of the input (e.g., pixels at the edge of an image) can be ignored or whether the memory required to store individual inputs can be reduced by, for example, zeroing the values of low-responsibility elements. Similarly, creators seeking to minimize a system's susceptibility to adversarial inputs could deploy input heatmaps to, for example, determine that the responsibility for final classifications should be distributed more evenly across all "input" elements. Of course, although creators may invoke input heatmaps to determine the particular modifications that must be made, they will normally have to use other techniques—in particular, techniques capable of answering how-questions (Sect. 5 below)—to actually implement those changes.

While input heatmapping can be used by stakeholders hoping to better understand *what* a system is doing, this technique can also be used by stakeholders interested in knowing *why* the system does what it does. Recall that why-questions are answered by specifying a regularity or correlation *f'* that obtains between features of the environment and by showing that this correlation is tracked by the system's transition function *f*. Input heatmaps can be used to answer why-questions if the environmental features that participate in *f'* can be discerned by inspecting the map—that is, if the highlighted elements of the "input" state together *look like* some recognizable feature of the environment. Consider again the input heatmaps in Fig. 2. The highlighted pixels visually resemble handwritten digits. In particular, the highlighted pixels in the upper-left heatmap visually resemble a handwritten two rather than, for example, a handwritten seven. This fact answers a question about *why* the upper-left input image outputs "2" rather than "7": the output is "2" *because* the image depicts a two. Stated more generally, the input heatmaps in Fig. 2 show that the system does what it is in fact supposed to do, namely, detect and classify handwritten digits.

In order to better appreciate the importance of answering why-questions in this way, it is worth contrasting the above example with one in which the computing system does *not* in fact do what it is supposed to do. Consider a well-known historical (albeit probably apocryphal) example in which a neural network learns to visually distinguish enemy tanks from friendly ones.[12] Although the network quickly learns to categorize images of tanks, it does so by tracking an accidental correlation between tank allegiances and weather patterns: whereas the images of friendly tanks were all taken on a sunny day, the enemy tanks were photographed under cloud cover. For this reason, although the system correctly classifies images (*what*), its reasons for doing so (*why*) have nothing at all to do with tanks!

In this example, heatmapping techniques such as LRP are likely to produce visualizations in which the highlighted pixels together resemble clouds in the background, rather than tanks in the foreground. Several different stakeholders would be likely to benefit from such visualizations: executors at military HQ who must decide whether or not to shoot at a particular tank, but also examiners at the International Criminal Court tasked with determining whether the resultant action was in fact a war crime. Moreover, tank crews who make up the decision subjects of the tank classification system could rest easy in their knowledge that the system can be trusted and that they are for this reason less likely to perish in a data-driven barrage of friendly fire.

Before moving on to questions beyond *what* and *why*, it is important to mention a significant limitation of the heatmapping technique. Input heatmaps can be used to answer why-questions when the highlighted elements together *look like* some recognizable feature of the environment. As was already suggested previously, however, Machine Learning methods are renowned for their ability to identify and track subtle as well as complex features of the learning environment, many of which may not be easily recognized or labeled by human observers (Buckner 2018). In such cases, the utility of input heatmapping is likely to be limited, and other XAI techniques may have to be invoked. One such technique might be *Local Interpretable Model-Agnostic Explanation* (LIME, Ribeiro et al. 2016), which can be used to simplify the transition function f (which is often nonlinear and therefore difficult to interpret) with a linear approximation f_{approx} that may be more readily interpreted by human observers. Similarly, *Local Rule-Based Explanations* (LORE, Guidotti et al. 2018) are designed to approximate limited domains of f by sets of comprehensible decision rules. Although these approximations always only capture a system's behavior for a limited range of inputs, and although as approximations they always bear the risk of oversimplification and

[12] Gwern Branwen maintains a helpful online resource on this particular example, listing different versions and assessing their probable veracity: https://www.gwern.net/Tanks (retrieved January 25, 2019).

misrepresentation, they might nevertheless prove useful for answering what- and why-questions when input heatmapping fails.

5 Intervention: The Algorithmic and Implementational Levels

5.1 Questions About How and Where

In Marr's account of explanation in cognitive science, the levels "below" the computational level of analysis are the *algorithmic* and *implementational levels*. The algorithmic level centers on questions about *how* a system does what it does (McClamrock 1991). Insofar as the system's behavior is described using a transition function *f* from an "input" state to an "output" state, the algorithmic level aims to uncover the mediating states $s_1, s_2,...s_n$ and state transitions $s_i \rightarrow s_j$ that appropriately connect "input" and "output" (Fig. 3). Put differently, the algorithmic level is concerned with uncovering the *program* that executes the overall transition *f* to thereby compute or approximate *f'* (Shagrir 2010).[13]

In contrast, the *implementational level of analysis* centers on questions about *where* the program described at the algorithmic level is realized (Zednik 2017). Where-questions concern the physical components $p_1, p_2,...p_m$ in which states are realized and state transitions are performed (Fig. 3). Thus, the implementational level is concerned with the hardware components involved in executing the program for *f*.

Whereas why-questions must be answered by "looking out" at a system's surrounding environment, how- and where-questions can only be answered by "looking inside" at the functional or physical variables, structures, and processes that intervene between the system's inputs and outputs.[14] For this reason, these questions are particularly important for the purposes of *intervening* on a system's behavior. Knowledge of a state s_i or a transition $s_i \rightarrow s_j$ that mediates the overall transition between "input" and "output" can be used to influence that overall transition by changing either one of s_i or $s_i \rightarrow s_j$. Likewise, knowledge of the fact that either s_i or $s_i \rightarrow s_j$ is physically realized in some physical structure p_k can be used to achieve the same goal by, for example, replacing p_k with some other physical structure p_l or by removing p_k altogether.[15]

In order to better understand the way in which creators should go about answering how- and where-questions in Explainable AI, it is helpful to first consider the way these questions are answered in cognitive science. There, answers to how-questions are typically delivered by developing *cognitive models* which describe the processes that govern a particular system's behavior. Notably, the relevant processes are only rarely described in terms of "brute causal" interactions between neuronal structures. More

[13] The program that mediates between "input" and "output"—the *learned program*—must not be confused with the learning algorithm that is used to develop (i.e., *to* program) the system in the first place.

[14] Intuitively, answers to how- and/or where-questions specify the EREs that are causally relevant to the behavior that is being explained. Although there are longstanding philosophical questions about the particular kinds of elements that may be considered causally relevant, the present focus on intervention suggests a maximally inclusive approach (see also Woodward, 2003).

[15] Although there is a clear sense in which interventions can also be achieved by modifying a system's inputs—a different s_{in} will typically lead to a different s_{out}—interventions on the mediating states, transitions, or realizers are likely to be far more wide-ranging and systematic.

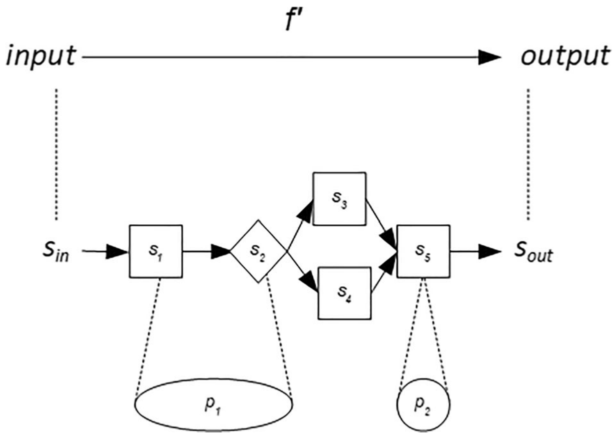

Fig. 3 As in Fig. 2 s_{in} and s_{out} are the “input” and “output” states of the computing system, respectively; input and output are the corresponding features of the environment; the dotted arrow designates a representation (or other kind of correspondence) relationship between the system and its environment. The squared shapes and arrows in the middle designated the individual states and state-transitions that are performed to compute or approximate *f,* the description of which answer a how-question. The circles at the bottom indicate the physical realizers of invidual states and/or state transitions, the identification of which answers a where-question.

commonly, they are described in terms of the numeric calculation of values or the stepwise transition between states (Busemeyer and Diederich 2010). The most important advantage of such descriptions is that they are able to capture a system’s abstract mathematical properties. However, in some cases, they also afford relatively straightforward semantic interpretations (Fodor 1987). That is, certain variables or states of a cognitive model might be said to represent specific features of the environment so that changes in those variables or transitions between those states capture changes in the things being represented. That said, representational interpretations may not always be forthcoming—for example, because it is unclear which features of the environment are actually being represented (Ramsey 1997)—nor useful, in particular, when the nonrepresentational description is deemed sufficiently useful for the purposes of interpreting, predicting, and intervening on the system’s behavior (Chemero 2000).

In turn, where-questions in cognitive science are typically answered by *localizing* the elements of a cognitive model—i.e., its individual states, state transitions, variables, or calculations—in specific physical structures such as neurons, neural populations, or brain regions (Piccinini and Craver 2011; Zednik 2017). Although cognitive scientists had long denigrated the explanatory relevance of where-questions (Pylyshyn 1984), it is now widely acknowledged that a proper understanding of the brain is critically important for the purposes of explaining the mind (Shallice and Cooper 2011). That said, it would be a mistake to think that localization in cognitive science is typically “direct” in the sense of affording simple one-to-one mappings between the individual steps of a process and the parts of an underlying physical structure (Bechtel and Richardson 1993). Indeed, the functional boundaries between the elements of cognitive models frequently cut across the physical boundaries between physical structures in the brain (Stinson 2016). Although this should not be taken to indicate that the answering of where-questions is impossible or unimportant, it does show that close attention should be paid to the sense in which, for example, the states and state transitions of a particular process—and thus, possibly, the corresponding representations and representation-manipulations—might be distributed (Clark 1993).

5.2 Creators

How might this brief foray into the explanatory norms and practices of cognitive science be relevant in the context of Explainable AI? Like scientists working to explain the behavior of biological cognizers, the creators of ML-programmed computing systems are regularly concerned with questions about *how* and *where*. On the one hand, software developers and system administrators tasked with developing, maintaining, fixing, and generally improving a system's behavior need to know not only *what* that system does but also *how* it does it. On the other hand, hardware engineers charged with building and maintaining the hardware in which the system is implemented must know *where* certain processes could be or are localized. Notably, much about the way in which how- and where-questions are answered in cognitive science can be used to better understand the way these questions can and should be answered in Explainable AI.

Consider how-questions first. Just as in cognitive science it is only rarely useful to look for "brute-causal" interactions between neuronal structures; in the Machine Learning context, it is only rarely useful to cite the values of individual learnable parameters such as a neural network's unit activations or connection weights. Indeed, as has already been discussed in Sect. 2, their high-dimensional complexity makes many ML-programmed computing systems unpredictable even with complete knowledge of the underlying parameter values. Moreover, there is often no way of knowing in advance whether an intervention on a single parameter will change the relevant system's behavior entirely or else affect it in a way that is mostly or entirely imperceptible.

Like in cognitive science, therefore, suitable answers to how-questions in Explainable AI will in most cases describe abstract mathematical properties of the system whose behavior is being explained. To better understand the particular kinds of properties that might be sought, consider once again the loan risk assessment system from above. In order to track statistical correlations between previous applicants' personal data and their ability to repay loans, the risk assessment system is unlikely to categorize new applicants on the basis of simple linear combinations of inputs such as age and income. Rather, the system is more likely to deploy a taxonomy of abstract categories in which the inputs are combined nonlinearly (see also Buckner 2018). For illustrative purposes, it can help to assume that these categories approximately correspond to folk-psychological character traits such as *honest*, *unscrupulous, high self-control, persevering,* or *foolhardy*. Although an applicant who once was the victim of a Ponzi scheme may appear inconspicuous to a bank employee, the automated risk assessment system might, through a nonlinear combination of data points, nevertheless classify that applicant as being foolhardy, and for this reason, risky. In order to properly understand *how* this system does what it does—and thus, in order to potentially intervene on the system's behavior—creators would almost certainly benefit from identifying the system's learned categories and from characterizing the role these categories play in the generation of particular outputs.

Now consider where-questions. Recall that questions of this kind were long denigrated in cognitive science. Because many ML applications are driven by standard-issue hardware components, it may be tempting to dismiss where-questions as being similarly irrelevant in Explainable AI. But it is important to resist this temptation. Indeed, a growing number of ML applications are driven by ML-specific hardware components. For example, the artificial neural networks used for visual processing in self-driving cars are increasingly implemented on neuromorphic hardware devices that have considerable advantages with respect to speed and power consumption (Pfeiffer and Pfeil 2018). In this sense, where-questions are hugely important at least for creators tasked with choosing the hardware in which to implement a particular computing system. But where-questions can also be important for creators tasked with maintaining, repairing, or optimizing a computing system once it has been built and deployed. Knowing the physical location in which certain kinds of data are stored and processed can allow creators to selectively replace hardware components so as to improve the speed or efficiency of the system as a whole. Moreover, in certain scenarios, it may even be important to know where data are processed *after* the system has stopped working. For example, in the aftermath of a fatal accident involving a self-driving car, it may be necessary to extract the data that was processed in the moments immediately preceding the accident, even if the system is no longer operational.[16] In such scenarios, a creator's ability to answer questions about *where* certain operations were performed may be instrumental for answering further questions about *what* the system was doing, as well as about *why* and *how*.

Surprisingly perhaps, where-questions may sometimes even be important to stakeholders other than creators. Although many computing systems are colloquially said to be realized "in the cloud," what is actually meant is that they are implemented in a device (or a cluster of devices) that is physically far-removed from the stakeholders in the relevant ecosystem. Given the differences in the ways different countries regulate the storage of information, data subjects and decision subjects may want to know the particular jurisdiction under which—i.e., *where* in the world—their data is being stored and processed. In a related way, examiners may be tasked with developing and enforcing legislation that limits the extent to which data can be exported. Thus, whereas where-questions are mostly within the purview of creators, there are situations in which these questions may even become important to other kinds of stakeholders.

5.3 Feature-Detector Visualization

One technique for answering how- and possibly even where-questions involves the identification and visualization of *feature detectors*. Just as what- and why-questions can be answered by highlighting features of the input that bear a high responsibility for the production of certain outputs, how-questions can sometimes be answered by

[16] Curiously, in such scenarios, a computing system's hardware components become analogous to the "Black Box" voice recorders used on commercial airliners.

highlighting those features of the input that are most responsible for activity in certain mediating variables. Insofar as the relevant variables are *sensitive* to a particular feature (i.e., respond reliably when the feature is present), relatively *unique* (i.e., no other mediating variables are similarly sensitive), and *causally efficacious* (i.e., they significantly influence the system's overall output), those variables can be viewed as *feature detectors*. Identifying a system's feature detectors—assuming there are any—and characterizing their influence on the system's overall behavior serves well for the purposes of answering questions about *how* that system works and sometimes even about *where* the relevant feature-detecting operations are carried out.

Consider a recent study due to Bau et al. (2018). This study considers *generative adversarial networks* (GANs) capable of producing photorealistic images of scenes depicting, for example, Christian churches (Fig. 4a). The aim of the study is to identify feature detectors and explore the systematic effects of surgical interventions on these detectors.[17] To this end, Bau et al. "dissect" the relevant networks to identify the units or unit clusters that are sensitive and unique with respect to recognizable features such as

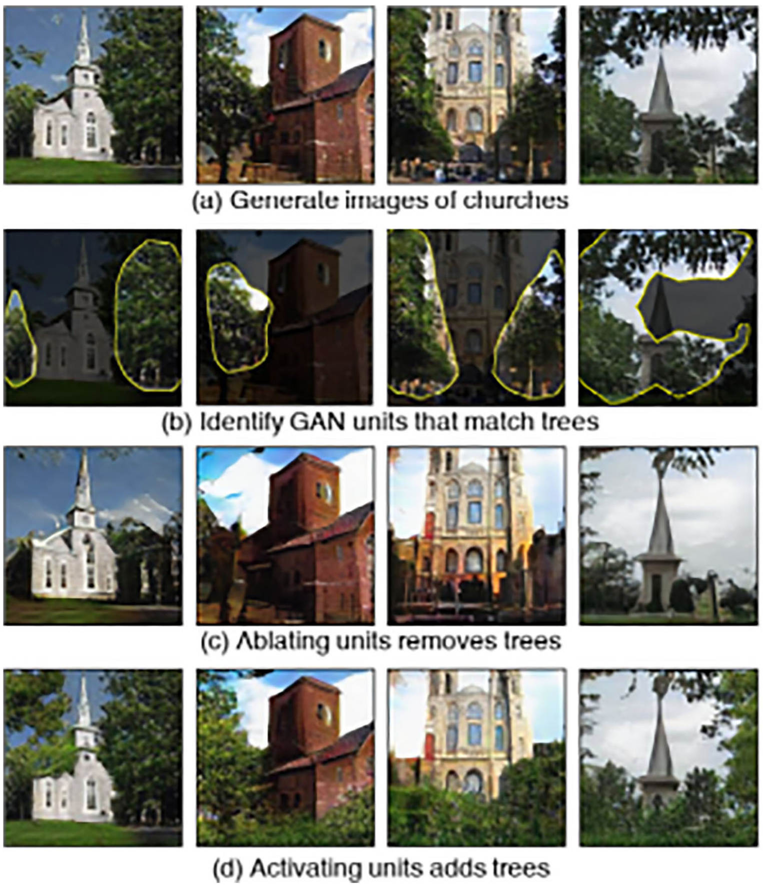
(a) Generate images of churches
(b) Identify GAN units that match trees
(c) Ablating units removes trees
(d) Activating units adds trees

Fig. 4 Images generated by a generative adversarial network: (**a**) generated images of christians churches and their surrounding; (**b**) features detected by the feature detector for trees; (**c**) images generated after ablating the feature detector for trees; (**d**) images generated after activating the feature detector for trees. Reproduced from Bau et al. 2018.

trees (Fig. 4b). Subsequently, they determine the causal efficacy of these (clusters of) units by performing a series of interventions such as activating or ablating (i.e., deactivating) the relevant units. Indeed, through these kinds of interventions, the authors are able to systematically control the presence or absence of features in the generated images, without compromising the overall image quality (Figs. 4c and 4d).

Bau et al.'s study exhibits many of the hallmarks of well-answered how-questions. For one, it shows that interventions on feature detectors can be used to repair or otherwise improve the relevant system's performance—a typical task for creators. For example, by identifying and subsequently ablating the feature detectors for unwanted visual artifacts such as textured patterns where there should be none, they are able to successfully remove those artifacts from the generated images and therefore improve the system's overall performance. For another, most of the features being detected by the networks in the study are robust with respect to nuisance variations in color, texture, and spatial orientation. That is, the GANs appear to learn just the kinds of high-level representations that one would expect to find in high-dimensional complex systems that are programmed using Machine Learning. Indeed, the authors even advance the hypothesis that the networks' representations resemble the conceptual representations that are used by human brains.

As this example shows, the identification of feature detectors is an effective technique for answering how-questions in the ML ecosystem. That said, it is worth considering the extent to which this technique might also be used to answer where-questions. Indeed, insofar as network's feature detectors are concentrated in a relatively small number of units and those units might be implemented in neuromorphic hardware components, the identification of feature detectors will answer questions about *how* and *where* simultaneously. Indeed, in such cases, modifications to the networks' overall behavior could equally be achieved by, for example, removing or replacing the hardware components that implement specific detectors.

5.4 Diagnostic Classification

Feature detectors can be invoked to answer questions about *how* and even *where,* but only when the responsibility for generating particular outputs is concentrated in a relatively small number of system variables (e.g., a small number of network units). In contrast, they cannot normally be invoked when the responsibility is distributed across a large number of variables (e.g., a layer or network as a whole). Indeed, decades-long discussions of connectionist modeling methods in cognitive science suggest that neural networks and similarly high-dimensional systems are very likely to exhibit this kind of distributed responsibility (Smolensky 1988), and thus, are likely to deploy distributed representations (Clark 1993). For this reason, investigators in Explainable AI have good reason to develop alternative techniques that can be used to answer how-questions even when no clearly circumscribed feature detectors can be identified.

Consider a recent study from computational linguistics, in which Hupkes et al. (2018) explore the capacity of different networks to evaluate expressions with nested

[0] Strictly speaking, because the aim of the GANs in this study is not detection but *generation*, the relevant units might more appropriately be called feature *generators*.

grammatical structures. It has long been known that *simple recurrent networks* (SRNs, Elman 1990), like other networks with recurrent feedback connections, perform well when confronted with tasks of this kind. What remains unclear, however, is exactly how these networks do what they do and, in particular, how they store and deploy information over extended periods. In particular, assuming that a nested arithmetic expression such as "(5-((2−3) + 7))" is processed from left to right, some symbols encountered early (e.g., the "5" and the leading "-") will have to be evaluated only after other symbols are encountered later.

Hupkes et al.'s challenge is to determine whether networks capable of evaluating such nested expressions do so by following either one of two strategies: a *recursive* strategy in which parenthetical clauses are evaluated only after they have been closed (2 −3 = −1; −1 + 7 = 6; 5−6 = −1) or a *cumulative* strategy, in which parentheses are removed from left to right while appropriately "flipping" the operators between "+" and "-" (5−2 = 3; 3 + 3 = 6; 6−7 = −1). In order to answer this question about *how* the relevant networks do what they do, Hupkes et al. deploy *diagnostic classifiers*: secondary networks that take as inputs the primary network's hidden unit activations while a particular symbol is being processed, and that generate as output a value that can be compared to a prior hypothesis about the information that *should* be represented at that moment. In the present example, the diagnostic classifiers' outputs are compared to the information that should be represented if the recurrent network was to follow either one of the relevant strategies. For example, after processing the "3" in the arithmetic expression "(5-((2−3) + 7))," a recurrent network that adheres to the recursive strategy should represent an intermediate sum of −1, whereas one that follows the cumulative strategy should represent 6. Indeed, Hupkes et al. find that the diagnostic classifiers generate outputs more in line with the cumulative strategy than with the recursive strategy (Fig. 5).

Like feature-detector identification, diagnostic classification can be used to answer questions about *how* a particular system does what it does. More precisely, this technique can be used to determine which information is represented by a system when it receives a particular input and, thus, how that network processes information as the inputs change over time. In a sense, therefore, diagnostic classifiers can be used to trace the particular computer program—in this case understood as a series of transitions between information-bearing states—that is being executed by networks capable of solving complex problems in AI.

Diagnostic classification has at least one important advantage over techniques that center on feature detectors, but also some significant disadvantages. On the one hand, diagnostic classifiers do not require that the representations in the relevant system be contained in a small number of variables. For this reason, they appear well-suited for answering how-questions even when—as is often likely to be the case—networks solve AI problems by manipulating distributed representations. On the other hand, diagnostic classifiers may be thought to be of comparatively limited explanatory value insofar as they do not readily afford the ability to *intervene*. Whereas feature detectors can be ablated or activated, and the resultant effects can be recorded, it is not clear how a diagnostic classifier's outputs can be used to systematically modify a system's overall behavior. Moreover, whereas in certain cases feature detectors may be cited to answer where-questions in addition to answering how-questions, diagnostic classifiers are

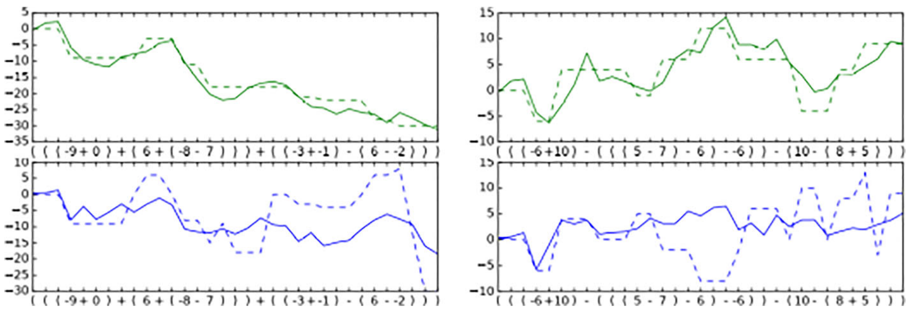

Fig. 5 Diagnostic classifier outputs (y-axis) for the cumulative (top) and recursive (bottom) strategies over the course of the two distinct expressions (x-axis). The classifiers' outputs (solid line) are compared to the prior hypothesis (dashed line), revealing an overall closer fit to the cumulative than to the recursive strategy. Reproduced from Hupkes et al. (2018).

unlikely to address the former other than by indicating that certain representations are realized somewhere within the system as a whole.

6 Conclusion

This discussion has sought to develop a normative framework for Explainable AI—a set of standards that should be satisfied for the purposes of rendering opaque computing systems transparent. Beginning with an analysis of "opacity" from philosophy of science, it was shown that opacity is agent-relative that that rendering transparent involves acquiring knowledge of epistemically relevant elements. Notably, given their distinct roles in the Machine Learning ecosystem, different stakeholders were shown to require knowledge of different (albeit equally "real") epistemically relevant elements. In this sense, although the opacity of ML-programmed computing systems is traditionally said to give rise to *the* Black Box Problem, it may in fact be more appropriate to speak of *many* Black Box Problems–one for every stakeholder. Depending on who you are and how it is that you interact with an ML-programmed computer, that computer will be opaque for different reasons and must be rendered transparent in different ways.

Explainable AI is in the business of developing analytic techniques with which to render opaque computing systems transparent and, thus, to solve the Black Box Problem. By deploying such techniques, stakeholders in the ML ecosystem should be able to more effectively develop, operate, trust, examine, and otherwise interact with ML-programmed computers. But although many powerful analytic techniques have already been developed, not enough is known about when and how these techniques actually explain—that is, when and in which sense these techniques successfully render the relevant computing systems transparent. Here, the present discussion sought inspiration in cognitive science. As one of the most influential normative frameworks for evaluating the explanatory successes of analytic techniques in cognitive science, Marr's *levels of analysis* account was used to determine what it actually takes to satisfy the distinct explanatory requirements of different stakeholders in the ML ecosystem. By aligning these stakeholders with different kinds of explanation-seeking questions—questions about *what*, *why*, *how*, and *where*—and by specifying the kinds of EREs that should be identified in order to answer these questions, it was possible to develop a normative framework with which to evaluate the explanatory successes of specific analytic techniques.

Finally, by reviewing a series of illustrative examples from Explainable AI, it was argued that many opaque computing systems can in fact be rendered transparent in the sense required by different stakeholders. This review demonstrated that techniques such as input heatmapping may be able to answer operators' questions about *what* a computing system is doing, but to also to answer data subjects' and decision subjects' questions about *why* the system does what it does. In addition, it showed that techniques such as feature-detector visualization and diagnostic classification may be useful for answering creators' and examiners' questions about *how* the system does what it does and, in certain circumstances, even questions about *where*. Thus, Explainable AI appears to be well-equipped for answering the questions that are most likely to be asked by different stakeholders and, thus, for finding solutions to the many different Black Box Problems.

That said, the normative framework developed here not only shows that the analytic techniques being developed in Explainable AI are useful but also that they have certain characteristic limitations. As was the case for input heatmapping, the techniques of diagnostic classification and feature-detector-identification work relatively well when system's variables can be interpreted semantically—that is, when they can be thought to represent recognizable features of the environment. Indeed, feature detectors are informative only when the features being detected can be recognized by human observers, and diagnostic classifiers require investigators to already possess a detailed understanding of the programs that *might* be executed by the system whose behavior is being explained. As has already been suggested, however, there are reasons to believe that the environmental features being detected and the correlations being learned through the use of Machine Learning may be subtle and difficult to interpret. For this reason, although these XAI techniques are undeniably promising, there is also considerable room for improvement.

To what extent can this improvement be made? Here it may once again be worth seeking guidance in cognitive science. There, one long-term trend is the gradual recognition that semantic interpretability is not always necessary to explain the behavior of humans and other biological cognizers. Indeed, it is now far less widely assumed than before that the neuronal processes described by cognitive models should—or even can—be described in semantic terms (Chemero 2000; Ramsey 1997). For this reason, many cognitive models today instead deploy sophisticated mathematical concepts and analytic techniques for idealizing, approximating, or otherwise reducing the dimensionality of the systems being investigated–even if they do not thereby render these systems semantically interpretable. Although the use of these concepts and techniques may render folk psychological categories inapplicable, they might nevertheless be used to satisfy important explanatory norms such as description, prediction, and intervention.

In conclusion, it seems fair to wonder whether Explainable AI might similarly move away from semantic interpretability and toward idealization, approximation, and dimension reduction. On the one hand, Artificial Intelligence is in part an engineering discipline tasked with developing technologies that improve the daily lives of regular individuals. For this reason, it is subject to constraints that pure sciences are not. Whereas it may not be important for laypeople to understand the processes that underlie visual perception, it is important that they know why their loan applications are being rejected. On the other hand, just as society trusts scientists to possess expert knowledge that remains inaccessible to laypeople, it may accept that certain ML applications can only be explained by experts who are capable of deploying the most sophisticated XAI techniques. In this case, it may be necessary to rely on societal mechanisms—for example, the testimony of expert witnesses in a court of law—to ensure that an individual's rights are being protected, even when that individual cannot him- or herself come to know the relevant facts. Thus, although semantic interpretability may often be beneficial, it may not always be essential.

Acknowledgements This work was supported by the *German Research Foundation* (DFG project ZE 1062/4-1). The author would also like to thank Cameron Buckner and Christian Heine for written comments on earlier drafts. The initial impulse for this work came during discussions of the consortium on "Artificial Intelligence - Life Cycle Processes and Quality Requirements" at the *German Institute for Standardization* (DIN SPEC 92001). However, the final product is the work of the author.

References

Bau, D., Zhu, J.-Y., Strobelt, H., Zhou, B., Tenenbaum, J. B., Freeman, W. T., & Torralba, A. (2018). GAN dissection: visualizing and understanding generative adversarial networks. *ArXiv, 1811*, 10597.

Bechtel, W., & Richardson, R. C. (1993). *Discovering complexity: decomposition and localization as strategies in scientific research* (MIT Press ed.). Cambridge, Mass: MIT Press.

Bickle, J. (2006). Reducing mind to molecular pathways: explicating the reductionism implicit in current cellular and molecular neuroscience. *Synthese, 151*(3), 411–434. https://doi.org/10.1007/s11229-006-9015-2.

Buckner, C. (2018). Empiricism without magic: transformational abstraction in deep convolutional neural networks. *Synthese, 195*(12), 5339–5372. https://doi.org/10.1007/s11229-018-01949-1.

Burrell, J. (2016). How the machine 'thinks': Understanding opacity in machine learning algorithms. *Big Data & Society, 3*(1), 205395171562251. https://doi.org/10.1177/2053951715622512.

Busemeyer, J. R., & Diederich, A. (2010). *Cognitive modeling*. Sage.

Chemero, A. (2000). Anti-representationalism and the dynamical stance. *Philosophy of Science*.

Churchland, P. M. (1981). Eliminative Materialism and the Propositional Attitudes. *The Journal of Philosophy, 78*(2), 67–90.

Clark, A. (1993). *Associative engines: connectionism, concepts, and representational change*. MIT Press.

Dennett, D. C. (1987). *The Intentional Stance*. Cambridge, MA: MIT Press.

Doran, D., Schulz, S., & Besold, T. R. (2017). What does explainable AI really mean? a new conceptualization of perspectives. *ArXiv, 1710*, 00794.

Durán, J. M., & Formanek, N. (2018). Grounds for trust: essential epistemic opacity and computational reliabilism. *Minds and Machines, 28*(4), 645–666. https://doi.org/10.1007/s11023-018-9481-6.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science, 14*, 179–211.

European Commission.(2016) *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*

Fodor, J. A. (1987). *Psychosemantics*. Cambrdige, MA: MIT Press.

Goodman, B., & Flaxman, S. (2016). European Union regulations on algorithmic decision-making and a" right to explanation". *ArXiv, 1606*, 08813.

Guidotti, R., Monreale, A., Ruggieri, S., Pedreschi, D., Turini, F., & Giannotti, F. (2018). Local rule-based explanations of Black Box Decision Systems. *ArXiv, 1805*, 10820.

Hohman, F. M., Kahng, M., Pienta, R., & Chau, D. H. (2018). Visual analytics in deep learning: an interrogative survey for the next frontiers. *IEEE Transactions on Visualization and Computer Graphics*.

Humphreys, P. (2009). The philosophical novelty of computer simulation methods. *Synthese, 169*(3), 615–626.

Hupkes, D., Veldhoen, S., & Zuidema, W. (2018). Visualisation and'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research, 61*, 907–926.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2017). Building machines that learn and think like people. *The Behavioral and Brain Sciences*.

Lipton, Z. C. (2016). The mythos of model interpretability. *ArXiv, 1606*, 03490.

Marcus, G. (2018). Deep learning: a critical appraisal. *ArXiv, 1801*, 00631.

Marr, D. (1982). *Vision: a computational investigation into the human representation and processing of visual information*. Cambridge, MA: MIT Press.

McClamrock, R. (1991). Marr's three levels: a re-evaluation. *Minds and Machines, 1*(2), 185–196.

Minsky, M. (ed) (1968). *Semantic Information Processing*. Cambridge, MA: MIT Press.

Montavon, G., Samek, W., & Müller, K.-R. (2018). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing, 73*, 1–15. https://doi.org/10.1016/j.dsp.2017.10.011.

Pfeiffer, M., & Pfeil, T. (2018). Deep learning with spiking neurons: opportunities and challenges. *Frontiers in Neuroscience, 12*. https://doi.org/10.3389/fnins.2018.00774.

Piccinini, G., & Craver, C. F. (2011). Integrating psychology and neuroscience: functional analyses as mechanism sketches. *Synthese, 183*(3), 283–311. https://doi.org/10.1007/s11229-011-9898-4.

Pylyshyn, Z. W. (1984). *Computation and cognition*. Cambridge, MA: MIT Press.

Ramsey, W. (1997). Do connectionist representations earn their explanatory keep? *Mind & Language, 12*(1), 34–66.

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.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Why should I trust you?: explaining the predictions of any classifier. *ArXiv, 1602*, 04938v3.

Rieder, G., & Simon, J. (2017). Big data: a new empiricism and its epistemic and socio-political consequences. In *Berechenbarkeit der Welt? Philosophie und Wissenschaft im Zeitalter von Big Data* (pp. 85–105). Wiesbaden: Springer VS.

Russell, S.J., Norvig, P. & Davis, E. (2010). *Artificial Intelligence: A Modern Approach* (3rd ed.). Upper Saddle River, NJ: Prentice Hall.

Shagrir, O. (2010). Marr on computational-level theories. *Philosophy of Science, 77*(4), 477–500.

Shallice, T., & Cooper, R. P. (2011). *The organisation of mind*. Oxford: Oxford University Press.

Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral and Brain Sciences, 11*(1), 1–23.

Stinson, C. (2016). Mechanisms in psychology: ripping nature at its seams. *Synthese, 193*(5), 1585–1614. https://doi.org/10.1007/s11229-015-0871-5.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. *ArXiv, 1312*, 6199.

Tomsett, R., Braines, D., Harborne, D., Preece, A., & Chakraborty, S. (2018). Interpretable to whom? a role-based model for analyzing interpretable machine learning systems. *ArXiv, 1806*, 07552.

Wachter, S., Mittelstadt, B., & Floridi, L. (2017). Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *International Data Privacy Law, 2017*.

Zednik, C. (2017). Mechanisms in cognitive science. In S. Glennan & P. Illari (Eds.), *The Routledge Handbook of Mechanisms and Mechanical Philosophy* (pp. 389–400). London: Routledge.

Zednik, C. (2018). Will machine learning yield machine intelligence? In *Philosophy and Theory of Artificial Intelligence 2017*.

Zerilli, J., Knott, A., Maclaurin, J., & Gavaghan, C. (2018). Transparency in algorithmic and human decision-making: is there a double standard? *Philosophy & Technology*. https://doi.org/10.1007/s13347-018-0330-6.