Machine Learning models as Mathematics: interpreting explainable AI in non-causal terms

Stefan Buijsman

Forthcoming in Durán, J. & Pozzi, G. (eds.) Philosophy of Science for Machine Learning: Core Issues and New Perspectives. Springer Nature

Abstract

We would like to have a wide range of explanations for the behaviour of machine learning systems. However, how should we understand these explanations? Typically, attempts to clarify what an explanations for questions such as 'why am I getting this output for these inputs?' have been approached from the philosophy of science, through an analogy with scientific (and often causal) explanations. I show that ML systems are best thought of as noncausal, specifically mathematical objects. We should therefore interpret these explanations differently, through analogy with mathematical explanations. I show that this still allows us to use much of the same theoretical apparatus, and argue that the asymmetry of many of the standard ML explanations can be accounted for in virtue of the link these systems have with concrete implementations.

1 Introduction

Machine Learning (ML) systems are notorious for requiring explanations. We do not know why they give the outputs that they do, and we even do not know why machine learning systems are as accurate as they are. In response, a wide variety of tools has been developed to try to elucidate the outputs of ML systems (Das and Rad, 2020), and theories have been developed to explain the success of ML systems (Shwartz-Ziv and Tishby, 2017). There are feature importance methods that calculate which input features were the most important for a particular output (e.g. Ribeiro et al., 2016; Lundberg and Lee, 2017), counterfactual methods that calculate the nearest alternative input for which the ML system returns a different (desired) output (Karimi et al., 2020) and much more. In turn, this has put forward the more philosophical issue of what it would mean to explain the behaviour (in a broad sense) of ML systems. This chapter focuses on that question, in particular in the light of the mathematical aspects

of machine learning systems. As discussed in section 2, we can understand ML systems as implementing mathematical functions. We use optimization procedures (typically stochastic gradient descent) to fit this function to the data set that is available, with the aim to then classify, predict or produce new instances. All of this suggests that ML systems are abstract objects. Moreover, this idea of an ML system being linked to a mathematical function is also relied on by some of the explainability methods just mentioned. For example, Simonyan et al. (2013) highlight pixels for which the gradient of the implemented function is particularly steep, thus directly using the mathematical function approximated by ML systems in the XAI method.

I will dive into this mathematical aspect of ML systems throughout the chapter. It is noteworthy, however, because most philosophical accounts at the moment approach the interpretation of explanations of ML systems using theories of scientific explanation, leaving the abstract, mathematical, nature of machine learning aside. For example, the overview in (Beisbart and Räz, 2022, p.2) mentions for a sub-type of ML systems: "DNNs are mathematical models" but includes no accounts of XAI in terms of mathematical explanations. More concretely, Buijsman (2022) and Watson and Floridi (2021) both use the causal interventionist framework from Halpern and Pearl (2005) and Woodward (2003) to present an account of explanations of ML systems. Erasmus et al. (2021) apply four different accounts of scientific explanation (deductive-nomological, Inductive Statistical, Causal-Mechanical and New Mechanist) to the machine learning context. There is a strong tendency to appeal to accounts of explanation from the philosophy of science, even though these have been developed primarily for describing concrete phenomena.

At the same time, technical explainability tools suggest that an analogy of this kind is suitable. Biswas et al. (2022) and Geiger et al. (2023), based respectively on theoretical work by Buijsman (2022) and Beckers and Halpern (2019), are examples of concrete explainability tools that use causal inference techniques in order to provide answers to why-questions about the outputs of ML models. In short, applying accounts developed for scientific explanations to the case of ML systems is standard practice, and also seems to make practical sense. Yet, there is a tension with the abstract, mathematical, nature of ML systems.

In section 2 I discuss this tension in more detail, and argue that it is better not to think of ML systems in strict causal terms. However, I do think that we can make sense of interventions to ML systems, despite their being non-causal. Here, I am sympathetic to the intervention liberalism of Emmerson (2021) as well as to manipulationist accounts of mathematical explanations (Baron et al., 2020). However, with the lack of causation there is a question of explaining the asymmetry of explanations of ML systems. I tackle this question in section 3. Finally, if we view ML systems as mathematical, then it also makes sense to consider the wider literature on mathematical explanations. In section 4 I argue that this fruitfully applies to machine learning, as we can see for example explanations by constraints (Lange, 2018) for the behaviour of ML systems.

2 Abstraction and explanations of ML outputs

2.1 The status of ML systems

The starting point for this chapter is that ML systems are a kind of abstract object, that can be understood as a mathematical model (see Chapter 10). Machine learning systems are often introduced as such. As an example, Zhou (2021) introduces ML systems in the following setup. First, there is a data set $D = \{\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_n}\} \in \mathcal{X}$ consisting of d-dimensional vectors $\mathbf{x_i} = (x_{i1}; x_{i2}; \ldots x_{id}$. This data set is often labelled with outcomes $y \in \mathcal{Y}$, formally structured in a training set consisting of pairs $(\mathbf{x_i}, y_i)$.[1] This allows us to tackle, among other learning problems, prediction problems. This is the problem to establish a mapping $f : \mathcal{X} \rightarrow \mathcal{Y}$ by learning from the training set. To do so, we need a learning algorithm $\mathcal{L}$ (also known as an optimization procedure) which aims to minimize the error between the learned f and the input-output pairs of the data set. While we can dive into further mathematical details here, the point is that presentations of ML systems proceed typically in this fashion: we start with a set of ordered pairs, and through a mathematical procedure a function is fit to this set as best we can. And so, we can interpret ML systems as approximating functions. Indeed, there is a host of theoretical results showing that different classes of neural networks (a type of ML system) can act as universal function approximators (Hornik et al., 1989; Castro et al., 2000; Schäfer and Zimmermann, 2006), solidifying the idea that ML systems in general approximate mathematical functions. Looking at ML systems from this perspective, it seems that they are first and foremost mathematical in nature.

However, the tendency to think otherwise is not a surprising one. When reading Erasmus et al. (2021, p.845), they describe the situation as follows for a medical imaging algorithm (MAIS): "Feeding the mammogram image into the input layer of the ResNet18 [A type of ML system, SB] and the convolutional operations of each layer are causal processes, while the signals sent between the different nodes and layers are causal interactions. These causal processes and interactions lead to the output of the MAIS." And indeed, we enter the input on a specific computer, that will then execute step by step the algorithm until it displays the resulting outcome of the calculations. There are specific causal processes on the computer chip that instantiate these calculations. So, there certainly is a causal process going on here in the concrete application of the ML system, as operations are executed in order in a physical system. If we want to explain an ML system, can't we focus our explanations on those causal processes?

One answer is to point again to the idea that ML systems are approximating mathematical functions and thus that we want to gain insight into the behaviour of those approximated functions. That is different from wanting to understand the causal processes used in the function approximation. A second argument for why the causal processes are not what we should consider in XAI

[1] Unsupervised learning, which can for example aim to cluster the data set into groups that are relevantly similar, does not use labelling of this kind, but instead only uses the data set D.

comes from the debate on the ontology of software more generally. Machine learning systems are a type of software, and we want explanations for the outputs/behaviour of these software systems. In other words, the phenomenon that we want to explain is a type of software, and so if we want to know the ontological status of what XAI aims to explain then it makes sense to look at the ontology of software. The debates here also highlight the same tension between abstraction and concreteness, as for example Colburn (1999) defines software as a "concrete abstraction", because it always requires reference to a medium of description (a programming language) and a medium of execution (the hardware on which the software will run). Colburn reasons that this reference to concrete implementations where the software runs on physical hardware is essential, but that software is nevertheless an abstraction as programmers prefer to abstract away from implementation to consider software as something that will run on any machine.

Still, it is hard to make sense of this position that software is both abstract and concrete at the same time. Instead, more recent accounts such as those by Irmak (2012), Turner (2011) and Duncan (2017) have focused on software as an abstract object, but more specifically an abstract artifact. The idea here is that software has a function, and is created with that intended function in mind, thus making it a kind of artifact. However, software cannot be identified with any concrete implementations on particular computers, and comes with a standard of correctness (bugs in the code and faults in execution). Specifically, (Duncan, 2017, p.27) defines a software program as: "a specification that consists of one or more programming language instructions and whose concretization is embodied by an artifact that is designed so that a physical machine may read the concretized instructions." This entails that software is an abstract specification rather than a concrete implementation, but it has to be such that it links to concrete versions (e.g. physical memory states of a computer) that can be read and executed.

This idea of software as a specification is also found in the work of Primiero (2016), based on Floridi's Levels of Abstraction. There, the highest level of abstraction is the intention of the programmer, which we hope to satisfy through the design of an algorithm. "An algorithm is the abstract representation of a mathematical function required to fulfil a task" (Primiero, 2016, p.98). This algorithm is then implemented in a programming language, which denotes the instructions in machine code and finally controls electrical charges in the hardware. The software is viewed as a type of abstract object, and specifically a mathematical object, but guided by an intention and linked to concrete implementations. We should, then, consider that ML systems are (as a kind of software/algorithm) likewise abstract objects that are linked to concrete implementations but not identified with them. That means that we cannot, strictly speaking, talk about causes in the ML system even if there are causal links between the execution steps of the concrete implementations. As a result, direct applications of philosophical accounts of explanation such as the causal-mechanical account of explanation to ML systems fail.

In short, I see two complementary reasons why we should resist a causal/scientific

interpretation of explainable AI. First, ML systems are plausibly interpreted as mathematical structures that approximate functions. They are mathematically defined and analysed, and so we should interpret explainable AI as the endeavour to further illuminate the behaviour of these mathematical objects. Second, AI systems are implemented as pieces of software and explainable AI aims to offer insight into the behaviour of AI qua software that produces outputs. As software is plausibly interpreted as abstract (and mathematical) in nature, we should thus also for this reason consider explainable AI as an endeavour to explain the behaviour of abstract (mathematical) objects. On both counts, then, an interpretation of AI in strictly causal/scientific terms is inappropriate.

2.2 Interpreting explanations of machine learning systems

How, then, can we make sense of explanations of ML outputs? I hold that there is still a suitable dependence relation to be found in the case of ML systems, such that ML outputs depend on the inputs, and the function approximated by the ML system depends on the training data and optimization procedure. We can cash out this dependence in terms of counterfactuals or interventions (depending on how liberal one wants to be with talk of interventions, which on many accounts are restricted to causation; cf Emmerson (2021)). We can then get an account of ML explanations closely aligned with the philosophy of science-inspired accounts, specifically those related to the manipulationist account of Woodward (2003). Furthermore, it can help us make sense of the formal machinery used in explainability methods such as Biswas et al. (2022) and Geiger et al. (2023), which is currently presented as based on causation.

The basic intuition behind this is as follows: although the ML system is independent from the physical implementation, we still select the data set ourselves, we pick an optimization procedure and learning parameters (e.g. in the case of a neural network we pick the network architecture, hyperparameters and initialization of the network weights). All of these choices impact the resulting ML system: if we had chosen something else here, we could have gotten a different ML system (approximating a different function). Furthermore, once we have an ML system that is optimized on the data set, there is an intuitive sense in which the output depends on the input. If the input had been different, then the output could be different as well. Similarly, if the internal parameters of the ML system (e.g. an artificial neuron in the case of a neural network, or a decision tree in the case of a random forest) had been different, then the output could have been different even when the input is kept fixed. As a range of philosophers has suggested (Baron et al., 2020; Jansson and Saatsi, 2019; Reutlinger, 2018; Woodward, 2018), we can then extend the manipulationist account of explanation from a purely causal one to one about counterfactual dependence. Explanations display patterns of counterfactual dependence between the explanandum and the explanans and we can find these kinds of patterns in ML systems.

Counterfactual dependence is, of course, still difficult to make sense of in mathematics. 2 + 2 could not be 5, no matter what we do, and similarly

theorems and results of calculations are necessary truths. Still, Baron et al. (2020) suggest one way in which we can make sense of such counterfactuals even in a purely mathematical setting. They define the truth-conditions of these counterfactuals using a closeness-based semantics: "$A \boxdot\!\!\rightarrow B$ is true at a world ω iff some possible or impossible world in which both A and B are true is closer to ω than any possible or impossible world in which A is true and B is false, if there are any possible or impossible worlds in which A is true." (Baron et al., 2020, p.4) Impossible worlds are needed due to the impossibility that actual mathematical truths are false and the closeness of worlds is then determined based on similarity: if B is more similar to A than C is, then B is closer to A than C. This gives an account of counterfactuals that is applicable to mathematics (Baron et al. (2020) show as much by applying it to a number of mathematical proofs) and can be used to further specify the idea of counterfactual dependence in this setting.

Specifically in the case of ML systems there is then a range of different types of counterfactual dependence that we can be interested in when demanding explanations. Most technical explainability methods focus on the pattern of dependence of the output on the input. What would happen to the output were we to have a different input to the ML system? Counterfactuals on the closeness-based account would evaluate this in the expected manner: the outcome is simply that which would result from a different input to the actual ML system. This helps us to figure out what function the ML system approximates, and consequently to anticipate what it will output in new situations. In that sense, there is a clear analogy to causal manipulationist accounts: by figuring out how the outputs (counterfactually) depend on the inputs, we know what we need to do in order to obtain a different output.

Alternatively, we can also be interested in why the ML system approximates one function rather than another. Here, the dependence of the ML system on the learning parameters, optimization method and training data are the focus. While there are few technical tools that focus on these dependencies, computer scientists are well-aware of a number of these dependencies. For example, it is common knowledge that biases in the training data often get picked up by the ML systems that are trained on this data. In the case of facial recognition systems this was clear in terms of the accuracy of these systems. First, a big discrepancy in accuracy was observed between faces of people with light skin colours (for which the system worked well) and faces of people with darker skin colours (for which the system was far less accurate) (Lohr, 2022). Then, this discrepancy was diagnosed as the result of a bias in the data set: the data consisted primarily of faces of people with light skin colours. By fixing the representativeness of the data set, the discrepancy in accuracy has been significantly reduced. If we interpret this in terms of an explanation in response to a why-question we can think of it as follows: the question was 'why is there such a large difference in accuracy between skin tones, rather than a much smaller one?' and the answer can be formulated in terms of the presence/absence of dark skin tones in the data set. Adjust that (i.e. had there been more data points of this kind in the training data) and the discrepancy becomes smaller.

In line with the somewhat unusual status of software systems in general, we can see these counterfactuals as closely linked to concrete actions we can take. To evaluate what a change to the inputs of an ML system would do, the easiest route is to simply execute an implementation of the ML system on the alternative input. Likewise, if we want to know whether a change to the composition of the data set leads to a suitably different function approximation, often the simplest way of finding this out is to simply gather more data and run the training procedure on the new data set. That is quite different from the steps needed to evaluate the counterfactuals in pure mathematics that Baron et al. (2020) discuss. We still shouldn't interpret them as causal in nature, but the causal steps that we can take for specific implementations can help us in finding out the counterfactual dependencies for ML systems. As long as the specific implementation has no malfunctions (the code contains no bugs, the hardware operates properly), the answers will be the same for the implementation as for the abstract ML system. However, this qualification of 'no bugs, no malfunctions' is precisely the qualifier that leads us to assign a different status to the ML system, and thus to the conclusion that we shouldn't speak directly of causes in explanations of these systems.

I see a number of benefits to this particular treatment of explanations of ML systems. First, it sticks closely to the steps we actually take to try to explain various aspects of these systems (though for more global/theoretical explanations of ML systems, see section 4). We adjust inputs, parameters, training data and more in an effort to understand what works and what doesn't. A wide range of current explainability tools, from counterfactual methods to feature importance (which is based either on a calculated gradient indicating what will cause the biggest change in outputs or on a sample of the local neighbourhood of similar inputs), looks precisely at what the ML system would output were the inputs to be different. Second, it helps us make sense of the recent causal inference-inspired methods. These are based on the use of structural equation models, stating how different variables depend on one another. Usually they are reserved for causal dependence, but Baron et al. (2020) claim that they can also be applied to mathematics and the kind of counterfactuals considered here.

They don't give much of an explanation as to how we can make sense of structural equation models in a non-causal context, so it is worthwhile to explore that here. SEMs do not themselves require that the relation explored is that of causation, but of course we do need to have a relation with the right set of features. Starting with the list of conditions that Woodward (2003) gives for interventions we can see how many might apply to this counterfactual dependence within ML systems.

1. I causes X
2. I acts as a switch for all other variables that cause X. That is, certain values of I are such that when I attains those values, X ceases to depend on the values of other variables that cause X and instead depends only on the value taken by I

3. Any directed path from I to Y goes through X. That is, I does not directly cause Y and is not a cause of any causes of Y that are distinct from X except, of course, for those causes of Y, if any, that are built into the I $\rightarrow$ X $\rightarrow$ Y connection itself; that is, except for (a) any causes of Y that are effects of X (i.e., variables that are causally between X and Y) and (b) any causes of Y that are between I and X and have no effect on Y independently of X;
4. I is (statistically) independent of any variable Z that causes Y and that is on a direct path that does not go through X.

(Woodward, 2003, p.98)

If we remove the notion of 'cause' here and consider what happens in the proposed ML setting, how much remains? The first condition can be met in terms of the alternative situation being considered being precisely one where X (e.g. input or data set) is changed. Criterion number 2 can easily be satisfied even in the abstract setting: changing the input value, data set or even some internal part of the ML system's algorithm is done in such a way that this changed variable ceases to depend on exogenous variables. Condition 3 can, at least in the context of ML systems, also be guaranteed: we know that e.g. any change to the input will only affect the output by going through the function implemented by the ML system. Likewise, any change to the data set only changes the learned function via the optimization procedure. Finally, condition 4 is satisfied as long as we make sure that the change of input, data set etc. does not relate to other parts of the ML system. The remaining question is whether counterfactual dependence can indeed stand in for causation. The main worry here (and on any account that understands mathematical explanations using counterfactual dependence) is whether it is appropriately asymmetric. Do these dependencies go only in one direction? If so, then it seems that we have a situation where we can use structural equation modelling based on changes to parts of an ML system. And indeed, various accounts of mathematical explanation (I already mentioned Baron et al. (2020), but Gijsbers (2017) also presents a 'quasi-interventionist' account of mathematical explanation that looks at this kind of counterfactual dependence for the explanatory power of some proofs) However, that asymmetry in the counterfactual dependence is far from obvious and so has to be examined in the next section.

First, though, one may wonder whether this kind of counterfactual dependence is the only possible interpretation of ML explanations, now that we need to look at theories of mathematical rather than of scientific explanation. Causal-mechanistic (Salmon, 1984) and unificationist theories (Kitcher, 1989) are the prominent alternatives for scientific explanation, so can we find analogues for these accounts when it comes to mathematical explanations such as those discussed here? They are certainly harder to find in the literature, but Frans and Weber (2014) defend a mechanistic account of mathematical explanations based on the mechanistic account of Bechtel and Abrahamsen (2005). It should be noted, however, that while the inspiration is mechanistic it gets very close to

a Woodwardian account that accounts for explanations in terms of difference-makers. They specifically argue that a particular proof is explanatory by: "(a) identifying a dependency to be explained, (b) identifying entities, (c) substituting the notion of activities with the notion of difference-makers, and (d) show that these difference-makers are organized such that the truth of the theorem is established." (Frans and Weber, 2014, p.240) In the specific case of ML systems proofs are less applicable, as we do not explain a theorem but rather a (set of) output(s). Still, there are mechanistic accounts applied to ML systems (which we could interpret as a mechanism between inputs and outputs) and so the translation to the mathematical setting is worth exploring, with less emphasis on difference-making. One might argue that when explaining why a particular output is obtained from a particular input we are in fact after a mechanism sketch of the operations performed on the input in order to produce the output. Explainability methods that aim to capture the black box behaviour in decision trees (Nauta et al., 2021) or with structural equation models (Geiger et al., 2023; Wu et al., 2023) would be good candidates for such a view, as they aim to find easier to comprehend operations that yield similar (and of course ideally the same) results. Of course, some adjustments will be needed: the mechanisms are not actually causal and talk of activities as in Bechtel and Abrahamsen (2005) is also not quite fitting for the mathematical setting. Substituting something like 'operations' in its place, as these are performed on the inputs and then on successive hidden layers, suffices for the specific case of ML systems, but of course will not do for mathematical explanations in general. So, if we do need a substitution in terms of difference-making here, as Frans and Weber (2014) suggest, then ultimately there will be little to distinguish a mechanistic approach to the explanations of ML systems from a Woodwardian/interventionist approach.

Unificationism does seem to offer a different story. On such an account, what does the explaining is the unification of a range of phenomena under a single argument schema. Perhaps this can be done through identifying difference-makers (Bangu (2017) for example argues for a somewhat revised version of unificationism in which case causes can be seen as one type of unifier), but the pinpointing of difference-makers is not central to the account. So, we might for example accept explanations in this case that group together a large number of similar cases without telling us what happens should relevant conditions change. This seems to make the application to mathematics even simpler, as there is no need to figure out what interventions and difference-making mean when it comes to mathematical objects. As such, we can propose unificationism as another candidate theory for explanations of ML systems: these explanations should aim to group input-output pairs together under a unifying argument schema. The success of such an account of course depends on how plausible unificationism then is in this domain; Knowles (2021) has recently argued against unificationist accounts of mathematical explanations. With the space available here I won't go into that debate, the point is rather that the shift from interpreting ML as scientific to interpreting it as mathematical need not hamper our efforts to apply the familiar accounts of explanation. They do need some

adjustment though, especially when it comes to accounting for the asymmetry of ML explanations. I turn to that issue next.

3 The asymmetry of ML explanations

Mathematical equations are symmetrical, as opposed to causal relations, and so there is a question whether the counterfactual dependence in the mathematical case is appropriately asymmetrical. For example, it is (often, as techniques such as differential privacy aim to counter precisely this inference) possible to work out the data set on which the ML system was optimized even without having full access to the ML system (Shokri et al., 2017; Olatunji et al., 2021; Hu et al., 2022). Were we to know exactly what the initialization, optimization procedure and outcome were, then we can expect that the training data can be retrieved as well. In short, it seems that there is a problem (as with other mathematical explanations; cf. Lange (2021)) with the asymmetry of explanations. For while the counterfactual dependence may be symmetrical, there is an intuitive idea (reflected in XAI practices) that we should explain the output of an ML system in terms of the inputs, and the behaviour of the ML system in terms of the training data and procedure. What ensures that the counterfactual dependence discussed in section 2 will only ever go in this intuitive direction?

Baron et al. (2020), in a similar vein to Jansson and Saatsi (2019), argue that asymmetry is guaranteed on the counterfactual account in virtue of the indeterminacy of counterfactuals going in the 'unintuitive' direction. So, while a change in input to an ML system has a clear effect on the output, the idea is that a specific change in output can be realized in many different ways by changing the input. As a result, there is not one unique change in input that corresponds to this change of output. Hence, we do not have a unique (true) counterfactual from output to input, guaranteeing the asymmetry of this kind of counterfactual dependence. (Jansson and Saatsi, 2019, p.17) state the same though as: "Fixing the explanans variable to its actual value should fix the explanandum variable to its actual value".

Lange (2021) has recently criticized this approach to ensuring the asymmetry of mathematical/abstract explanations. He argues that a good number of causal explanations do not match this kind of asymmetry. For example, when considering that the window broke because the rock hit it, then there is no particular counterfactual that holds in the situation that the rock did not hit the window (other than that the window doesn't break). This particular indeterminacy need not be a problem, as both Baron et al. (2020) and Jansson and Saatsi (2019) focus only on whether the explanandum variable is fixed, and that is indeed the case here (as Lange admits, not throwing the rock will fix that the window does not break). However, it does seem to go the other way: if we're clear enough about the situation, it seems likely that the counterfactual 'had the window not been broken, then the stone would not have been thrown' is true in the closest possible world. After all, this requires by far the fewest changes; far fewer than, e.g. ensuring that the glass was bullet-proof and therefore does not

break when the stone hits. In addition, it brings us back to a symmetrical setup in terms of counterfactuals: had the stone not been thrown, the window would not have broken and had the window not broken, the stone would not have been thrown. Indeterminacy in one direction does not seem to offer a way out. Lange, in a second argument, echoes this sentiment: in the example of mother distributing 23 strawberries over 3 children we can reason just as well from her having given 7 strawberries to each child with 2 left over to the number of children and strawberries, and link adjustments to the result (her attempt at dividing) to what would then change to the initial situation (the number of children and strawberries). In the end, the indeterminacy seems to go away if we fill in sufficiently many details.

The case of ML systems further reinforces that issue. Here, we have a range of tools precisely designed to calculate the minimal change to the input in order to achieve a desired output known as counterfactual methods/algorithmic recourse (see Karimi et al. (2020) for a review). These methods do, for a given distance function over the inputs, return a unique alternative input that leads to the specified alternative output. In practice it is very difficult to create realistic distance functions, but this doesn't matter for the principled argument here. For these technical tools further solidify the claim that if we specify the situation and the notion of 'closer' enough then there will be true counterfactuals in both directions. So, the asymmetry of mathematical/abstract explanations is an issue when we understand such explanations on the basis of counterfactuals (and see Lange (2021) for further discussion of other attempts to ground the asymmetry).

The specific case of software has a serious benefit here in getting out of these difficulties. Inherent to software is an asymmetry in terms of execution: an earlier line of code/set of instructions is (in any implementation) to be executed before the next. We can apply this more broadly to ML systems: we know that the data set has to be there first, to then feed (step by step) into the optimization procedure and finally for the ML system to process an input line by line to reach an output. The specification of these steps may be an abstract object, but the asymmetry in terms of what happens earlier and what happens later (and thus which parts of the ML system can influence which other parts) is inherent. The values used in the first line of code cannot depend on the values used in the last line of code (at least, if we fully work out recursive functions for clarity's sake). That difference in dependence: we determine one value using another *in the execution of an ML system*, and not the other way around, is a candidate for the explanatory asymmetry.

This also aligns with the idea that an explanation displays a pattern of counterfactual dependence. In these implementations, a change to one of the earlier parts of the process (the data, the optimization procedure, the inputs) will affect the later parts of the ML system, but not vice versa. Changing the input will not affect the training data, and manually changing the output value (during the execution of the ML system) will not change the input of the system.

However, how can we best understand this dependence? We could leave it as is, and opt along with Lange (2021) for a kind of pluralism in where the asymmetry of non-causal explanations comes from. In the case of software, we

might say, the asymmetry is grounded in what has to be done first in any implementation of the ML system. We can rely on the asymmetry of causation in these concrete implementations, and be done with it. This account of asymmetry is not available in other cases of mathematical explanations, as these do not link to concrete implementations in the same way. Algorithms, as step-by-step procedures, are in that sense different from most other mathematical objects. Even staying with the case of ML systems we can see this difference, as the next section shows in a brief exploration of explanations by constraint of ML systems.

4 Explanations by constraint in ML systems

There are further explanations of the behaviour of ML systems that we should not forget. These are more closely related to the typical examples of mathematical explanations of mathematical facts, for example to the explanations by constraint presented in Lange (2018). One example, discussed in detail by Räz (2022), is the attempt to explain why deep neural networks (as a type of ML model) do not overfit despite being overparameterized. GPT-4, for example, is reported to have over a trillion parameters, which we would normally expect to promote literal memorization of the training data and thus very poor generalizability over new inputs. This does not happen, so why are DNN models so good at handling new data? Here, the Information Bottleneck (IB) theory developed by Shwartz-Ziv and Tishby (2017) offers one possible explanation. According to this theory, the success is the result of a trade-off between compression and prediction. This is a general trade-off that has to be made by prediction methods, but the argument goes that deep neural network lead to particularly good results on the trade-off; at or close to the theoretical (Pareto) optimum. Shwartz-Ziv and Tishby (2017) observe that in the initial training phases the DNN optimizes on predictive accuracy, storing more and more information about the input space in order to give the best possible results. After this, there is a second phase in the training where they observed that the stored information is then reduced (compressed), but in such a way as to minimize the loss of predictive accuracy. The optimization procedure thus searches for the information about the input space that is necessary to accurately predict the output, but the end goal is to keep only the minimally sufficient information to do so. Because of this second compression step DNNs do not overfit as much as the number of parameters would lead us to expect, as compression ensures that the irrelevant details of the training data are not used to determine the output. Furthermore, we can interpret this as a high-level explanation by constraint (the IB theory specifies a trade-off for any learning mechanism and DNNs make this trade-off in a particular way) as well as a counterfactual explanation (had the second, compression, phase not happened, DNNs would have overfit much more).

A second example is the impossibility theorems regarding fairness metrics for machine learning systems (Kleinberg et al., 2016). This theorem focuses

on three widely used statistical definitions of fairness between groups A and B: (1) equal probability of being assigned to the positive predicted class, (2) equal true positive rates, and (3) equal true negative rates. The impossibility theorem shows that except for some very special cases (e.g. equal base rates in groups A and B) it is impossible to satisfy all three of these definitions of fairness at the same time. Hence, there will be a trade-off between them. We can then use this theorem to explain why, for ML systems, we see that when an ML system is (perfectly) fair according to one definition, it is not (perfectly) fair according to the others. To cite on of their examples: "suppose we want to determine the risk that a person is a carrier for a disease X, and suppose that a higher fraction of women than men are carriers. Then our results imply that in any test designed to estimate the probability that someone is a carrier of X, at least one of the following undesirable properties must hold: (a) the test's probability estimates are systematically skewed upward or downward for at least one gender; or (b) the test assigns a higher average risk estimate to healthy people (non-carriers) in one gender than the other; or (c) the test assigns a higher average risk estimate to carriers of the disease in one gender than the other." (Kleinberg et al., 2016, p.17) So, when one wonders why we observe this trade-off in practice, the impossibility theorems can explain this fact in a principled manner. There is a constraint (specified by the theorem) on the properties that the approximated functions can have. Had this constraint not been there, presumably the situation would have been different.

We can make good sense of these mathematical explanations, as would be expected given their similarity to cases already discussed in the literature on mathematical explanation. However, the accompanying downside of this similarity is that it is far less plausible to appeal to the asymmetry inherent in the implementations of ML systems. These constraints, such as that it is impossible to meet two fairness criteria at the same time, do not relate to specific ML systems that can be implemented nor do they link to a procedure in which one part is executed before another. As a result, we cannot use the causal links in the implementation of an ML system to ground the asymmetry of these non-causal explanations. A pluralist interpretation of these (noncausal) explanations seems the most likely, at least in terms of our account of the asymmetry of the explanations.

It may be tempting to avoid this kind of pluralism through an appeal to a more general notion of 'grounding' here to capture both types of mathematical explanations. After all, there is a sense in which the outputs of an ML system are grounded (as in, determined by) the inputs as well as the training data and optimization procedure. Schaffer (2016) has proposed an understanding of metaphysical grounding using structural equation models, which provides a nice link to the explanations of ML outputs as discussed in section 2. Additionally, there have been suggestions to capture more standard mathematical (and conceptual) explanations in terms of grounding (Poggiolesi and Genco, 2023). On a general gloss of the situation we could then say that the asymmetry of explanations is accounted for in both cases through the idea that the explanandum is grounded in the explanans.

Such a move towards uniformity would be misleading, however. The conceptual grounding pushed by Poggiolesi and Genco (2023) relies on the use of conceptual complexity. The idea is that more complex concepts may be introduced (in the formal, proof-theoretic sense of having an introduction rule in a deductive system) on the basis of less complex concepts. In this sense, more complex concepts are grounded in less complex concepts and complexity is understood in terms of being further down the proof-theoretic chain. The fact that only introduction rules provide grounding helps to account for the asymmetry of the grounding relation and, similarly, the asymmetry of conceptual explanations. We cannot have explanations of simpler concepts in terms of more complex concepts (e.g. explaining A in terms of $A \wedge B$) because those inference require us to use elimination rules, and thus $A \wedge B$ is more complex than A. Here the difference with the input-output type explanations becomes clear again: such explanations involve calculations which, even if we could cash them out in terms of a full deductive system, will not generally conform to the idea that outputs are arrived at from inputs using introduction rules (if we strictly follow the Peano axioms then one will typically use $\forall$-elimination on $\forall x \forall y(x + S(y) = S(x + y))$ and similar axioms for other mathematical operations). But on a more basic level there is no clear reason why the conceptual complexity of an ML output (in this proof-theoretic sense) should be higher than the conceptual complexity of the ML inputs. Since ML systems often aim to predict, it may very well happen that values of a simpler (but harder to measure) concept are predicted using values of more complex concepts. We still want explanations in the XAI sense to explain how the output values depend on the input values, but these do not track conceptual complexity. ML systems may well be an exceptional case of noncausal explanations here, but the point remains that a unification (using at least these conceptions of grounding) only succeeds by obscuring the underlying differences.

5 Conclusion

Explanations of ML systems have typically been considered through the lens of scientific explanations. I have argued that ML systems are best seen as abstract (and plausibly mathematical) objects, to which many of the accounts of scientific explanation do not apply. However, similar accounts of mathematical explanation can be used to interpret explanations of ML systems and to make sense of the use of causal inference techniques in the case of ML systems. Counterfactuals of the form "had the input been x' instead of x, the output would have been y' instead of y" can still be evaluated even without causal relations. Furthermore, the changes that are made to values in an ML system meet all the criteria for interventions aside from the presence of causal relations, thus allowing us to interpret structural equation models in the ML case.

This does raise a worry, however, as without the presence of causality there is no guarantee that the account will be appropriately asymmetrical. Counterfactuals can go both ways, so how is the directionality of explanations ensured?

I argued that in the case of ML systems we can typically appeal to the direction inherent to the instructions of which an ML system is made up. To further strengthen this idea, we can also link counterfactuals to concrete implementations of the abstract ML system, where these instructions are executed with causal links in between them.

While that covers a wide range of explanations of ML systems, it fails to include a set of mathematical explanations that is not directly linked to concrete implementations. Explanations answering questions such as why deep neural networks do not overfit, or why we observe a trade-off between fairness metrics in ML systems, are better understood as explanations by constraint. These are still compatible with current counterfactual accounts of non-causal explanations, but there seems to be a big difference in how we will account for their asymmetry. Pluralism on this front, as also suggested by Lange (2021) based on a different set of mathematical explanations, seems the most plausible option.

References

Bangu, S. (2017). Scientific explanation and understanding: unificationism reconsidered. *European Journal for Philosophy of Science*, 7:103–126.

Baron, S., Colyvan, M., and Ripley, D. (2020). A counterfactual approach to explanation in mathematics. *Philosophia Mathematica*, 28(1):1–34.

Bechtel, W. and Abrahamsen, A. (2005). Explanation: A mechanist alternative. *Studies in History and Philosophy of Science Part C: Studies in History and Philosophy of Biological and Biomedical Sciences*, 36(2):421–441.

Beckers, S. and Halpern, J. Y. (2019). Abstracting causal models. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 2678–2685.

Beisbart, C. and Räz, T. (2022). Philosophy of science at sea: Clarifying the interpretability of machine learning. *Philosophy Compass*, 17(6):e12830.

Biswas, S., Corti, L., Buijsman, S., and Yang, J. (2022). Chime: Causal human-in-the-loop model explanations. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 10, pages 27–39.

Buijsman, S. (2022). Defining explanation and explanatory depth in xai. *Minds and Machines*, 32(3):563–584.

Castro, J. L., Mantas, C. J., and Benıtez, J. (2000). Neural networks with a continuous squashing function in the output are universal approximators. *Neural Networks*, 13(6):561–563.

Colburn, T. R. (1999). Software, abstraction, and ontology. *The Monist*, 82(1):3–19.

Das, A. and Rad, P. (2020). Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*.

Duncan, W. D. (2017). Ontological distinctions between hardware and software. *Applied Ontology*, 12(1):5–32.

Emmerson, N. (2021). A defence of manipulationist noncausal explanation: The case for intervention liberalism. *Erkenntnis*, pages 1–23.

Erasmus, A., Brunet, T. D., and Fisher, E. (2021). What is interpretability? *Philosophy & Technology*, 34(4):833–862.

Frans, J. and Weber, E. (2014). Mechanistic explanation and explanatory proofs in mathematics. *Philosophia Mathematica*, 22(2):231–248.

Geiger, A., Potts, C., and Icard, T. (2023). Causal abstraction for faithful model interpretation. *arXiv preprint arXiv:2301.04709*.

Gijsbers, V. (2017). A quasi-interventionist theory of mathematical explanation. *Logique et Analyse*, (237):47–66.

Halpern, J. Y. and Pearl, J. (2005). Causes and explanations: A structural-model approach. part i: Causes. *The British journal for the philosophy of science*.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Hu, H., Salcic, Z., Sun, L., Dobbie, G., Yu, P. S., and Zhang, X. (2022). Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)*, 54(11s):1–37.

Irmak, N. (2012). Software is an abstract artifact. *Grazer Philosophische Studien*, 86(1):55–72.

Jansson, L. and Saatsi, J. (2019). Explanatory abstractions. *The British Journal for the Philosophy of Science*, 70(3):817–844.

Karimi, A.-H., Barthe, G., Schölkopf, B., and Valera, I. (2020). A survey of algorithmic recourse: definitions, formulations, solutions, and prospects. *arXiv preprint arXiv:2010.04050*.

Kitcher, P. (1989). Explanatory unification and the causal structure of the world.

Kleinberg, J., Mullainathan, S., and Raghavan, M. (2016). Inherent trade-offs in the fair determination of risk scores. *arXiv preprint arXiv:1609.05807*.

Knowles, R. (2021). Unification and mathematical explanation. *Philosophical Studies*, 178(12):3923–3943.

Lange, M. (2018). Because without cause: Scientific explanations by constraint. *Explanation beyond causation*, pages 15–38.

Lange, M. (2021). Asymmetry as a challenge to counterfactual accounts of non-causal explanation. *Synthese*, 198(4):3893–3918.

Lohr, S. (2022). Facial recognition is accurate, if you're a white guy. In *Ethics of Data and Analytics*, pages 143–147. Auerbach Publications.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Nauta, M., Van Bree, R., and Seifert, C. (2021). Neural prototype trees for interpretable fine-grained image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14933–14943.

Olatunji, I. E., Nejdl, W., and Khosla, M. (2021). Membership inference attack on graph neural networks. In *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 11–20. IEEE.

Poggiolesi, F. and Genco, F. (2023). Conceptual (and hence mathematical) explanation, conceptual grounding and proof. *Erkenntnis*, 88(4):1481–1507.

Primiero, G. (2016). Information in the philosophy of computer science. In Floridi, L., editor, *The Routledge handbook of philosophy of information*, pages 90–106. Routledge.

Räz, T. (2022). Understanding deep learning with statistical relevance. *Philosophy of Science*, 89(1):20–41.

Reutlinger, A. (2018). Extending the counterfactual theory of explanation. *Explanation beyond causation: Philosophical perspectives on non-causal explanations*, pages 74–95.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). " 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*, pages 1135–1144.

Salmon, W. C. (1984). *Scientific explanation and the causal structure of the world*. Princeton University Press.

Schäfer, A. M. and Zimmermann, H. G. (2006). Recurrent neural networks are universal approximators. In *Artificial Neural Networks–ICANN 2006: 16th International Conference, Athens, Greece, September 10-14, 2006. Proceedings, Part I 16*, pages 632–640. Springer.

Schaffer, J. (2016). Grounding in the image of causation. *Philosophical studies*, 173(1):49–100.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE.

Shwartz-Ziv, R. and Tishby, N. (2017). Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Turner, R. (2011). Specification. *Minds and Machines*, 21:135–152.

Watson, D. S. and Floridi, L. (2021). The explanation game: a formal framework for interpretable machine learning. In *Ethics, Governance, and Policies in Artificial Intelligence*, pages 185–219. Springer.

Woodward, J. (2003). *Making things happen: A theory of causal explanation*. Oxford university press.

Woodward, J. (2018). Some varieties of non-causal. *Explanation beyond causation: Philosophical perspectives on non-causal explanations*, page 117.

Wu, Z., D'Oosterlinck, K., Geiger, A., Zur, A., and Potts, C. (2023). Causal proxy models for concept-based model explanations. In *International Conference on Machine Learning*, pages 37313–37334. PMLR.

Zhou, Z.-H. (2021). *Machine learning*. Springer Nature.