Ethics Inf Technol (2011) 13:251–260
DOI 10.1007/s10676-010-9233-7

Is there an ethics of algorithms?

Felicitas Kraemer · Kees van Overveld · Martin Peterson

Published online: 3 July 2010

Abstract We argue that some algorithms are value-laden, and that two or more persons who accept different value-judgments may have a rational reason to design such algorithms differently. We exemplify our claim by discussing a set of algorithms used in medical image analysis: In these algorithms it is often necessary to set certain thresholds for whether e.g. a cell should count as diseased or not, and the chosen threshold will partly depend on the software designer's preference between avoiding false positives and false negatives. This preference ultimately depends on a number of value-judgments. In the last section of the paper we discuss some general principles for dealing with ethical issues in algorithm-design.

Keywords Algorithm · Image analysis · Medical technology · False positive · False negative

Introduction

The focus of this article is on ethical aspects of algorithms. An algorithm is, roughly speaking, a finite sequence of well-defined instructions that describe in sufficiently great detail how to solve a problem. Both computers and humans use algorithms for solving a wide range of problems. However, in this paper we shall be exclusively concerned with algorithms implemented in computers.

At first glance it might be tempting to conclude that algorithms are value-free entities that do not, at least not in their most abstract form, have any ethical dimensions. However, in this article we argue that this commonsense view about algorithms is false. Many, but not all, algorithms implicitly or explicitly comprise essential value-judgments. By an 'essential value-judgment' we mean the following: If two algorithms are designed to perform the same task, such as classifying a cell as diseased or non-diseased, these algorithms are essentially value-laden if one cannot rationally choose between them without explicitly or implicitly taking ethical concerns into account. Another way of saying this is that the algorithm cannot be designed without implicitly or explicitly taking a stand on ethical issues, some of which may be highly controversial.

If true, our claim about essentially value-laden algorithms has to be taken seriously by software engineers who design algorithms. If some algorithms are essentially value-laden, i.e. if people who design algorithms cannot avoid making ethical judgments about what is good and bad, then it is reasonable to maintain that software designers are morally responsible for the algorithms they design.[1] Although the term 'ethics of algorithms' might have far-reaching connotations, it nevertheless captures what is at stake here. If our claim about essentially value-laden

F. Kraemer · K. van Overveld · M. Peterson (✉)
Section for Philosophy and Ethics, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
e-mail: m.peterson@tue.nl
URL: www.martinpeterson.org

F. Kraemer
e-mail: f.kraemer@tue.nl

K. van Overveld
e-mail: cvoverve@tue.nl

[1] Software designers can, of course, be morally responsible also for algorithms that are not essentially value-laden. A fatal accident caused by a faulty algorithm can be (partially) blamed on the software designer, irrespective of whether the algorithm is essentially value-laden or not.

algorithms is correct, there will indeed be an ethics of algorithms.

The issue that we address has to some extent already been touched upon in the literature. Turilli seeks to develop a solution to "the problem of specifying computational systems that behave in accordance with a given set of ethical principles".[2] Allen et al. argue that, "we need to integrate artificial moral agents into … new technologies to manage their complexity".[3] Finally, Arkin argues that there is a need for, "immediate investment in … machine/robot ethics".[4] Although we do not necessarily disagree with these authors, we wish to point out that their papers are concerned with questions that are slightly different from the issue discussed here. Turilli, Allen et al., and Arkin discuss how we can construct algorithms and systems that behave in *accordance with* ethical principles. As we argue in the next section, such algorithms need not be essentially value-laden themselves. Indeed, an algorithm or system that behaves in *accordance with* some ethical principles can itself be ethically neutral. What we wish to discuss in this article is the ethical features of the algorithm itself.

The structure of this article is as follows. In "Value-laden algorithms and the design process" we characterise some of the central concepts used in the paper and argue that algorithm design is in many respects similar to designing other technological artefacts. In "A real example" we discuss a real example of an essentially value-laden algorithm used in medical image analysis, and in "The precautionary principle and ethical theories" we discuss some ways of dealing with the ethical problems raised by this algorithm. Finally, in "Should ethical values be user-defined?", we argue that software designers should as far as possible leave ethical choices to users, and when this is not possible the ethical assumptions underlying the algorithm should at least be made transparent. This may lead to new or refined procedures in the process of software engineering.

Value-laden algorithms and the design process

It is notoriously difficult to give a precise characterisation of what an algorithm is. Numerous definitions have been proposed in the literature.[5] However, nothing in our reasoning hinges on how exactly algorithms are formally defined. The definition of an algorithm suggested in "Introduction" will suffice for our present purposes.

Arguably, the claim that many algorithms implicitly or explicitly comprise essential value-judgments cannot be accurately assessed without first defining some of its central concepts. We take a value-judgment to be any proposition expressing a view on how things ought to be or not to be, or what is good or bad, or desirable or undesirable. It takes little reflection to see that not all algorithms express value-judgments. Consider, for instance, algorithms used for calculating the lexicographical order of a finite set of strings, such as the ones used to put words in alphabetical order. Given the input to this algorithm, there is only one possible correct output. Therefore, this task is fully specified in terms of various logical relationships, which require no approximation or interpretation.

That said, some algorithms clearly produce genuine value-judgments. Consider, for example, algorithms used in decision support programs, i.e. systems that help decision makers to make better decisions by ranking a set of alternative actions with respect to some predefined criteria. A typical outcome of an algorithm used in such a program is a verdict like "Alternative X is the best option" or "Alternative X is better than alternative Y with respect to criterion Z". It would be pointless to deny that these sentences express genuine value-judgments. This is acknowledged by Turilli, who discusses how to specify "systems that behave in accordance with a given set of ethical principles".[6] However, the fact that an algorithm or system yields a value-judgment as its output does not prove that the algorithms used for producing the value-judgments are essentially value-laden, as pointed out in "Introduction". The string of sentences (or data) that is produced by running through an algorithm might very well be value-laden even if the algorithm itself is not value-laden.

Let us take a closer look at this important point. The general features of the kind of value-judgments generated by decision support programs can be clarified by invoking Kant's famous distinction between hypothetical and categorical imperatives. A hypothetical imperative is a statement about what you ought to do, given that you wish to respect some exogenously defined goal. "Don't study philosophy if you wish to become rich" is a hypothetical imperative. A categorical imperative is a value-judgment that tells us what we ought to do irrespective of our desires or goals. The distinction between hypothetical and categorical imperatives does not exhaust the logical space of all possible value-judgments, but it illuminates the nature of the kind of value-judgments that are most likely to be encountered in computer programs used for aiding

[2] Turilli (2007: 49).

[3] Allen et al. (2006: 13).

[4] Arkin (2009: 2).

[5] See e.g. Sipser (1997).

[6] Turilli (2007: 49).

decisions. Such value-judgments are best characterised as hypothetical value-judgments, because they tell you what you ought to do given that you wish to achieve a specific aim. However, as pointed out above, the presence of a hypothetical value-judgment does not entail that the algorithm used for generating the recommendations comprises value-judgments. The underlying algorithm can, when taken in isolation, be completely value-free.

So what, then, could it possibly mean to say that an algorithm comprises a value-judgment and therefore has an essential ethical dimension? Consider the following suggestion.

> ALGORITHM COMPRISING AN ESSENTIAL VALUE-JUDGMENT
> An algorithm comprises an essential value-judgment if and only if, everything else being equal, software designers who accept different value-judgments would have a rational reason to design the algorithm differently (or choose different algorithms for solving the same problem).

Note that this criterion does not make the definition circular. It is indeed true that the term 'value-judgment' occurs both in the left-hand part and in the right-hand part of the criterion, but the term that is to be explained is 'algorithm comprising an essential value-judgment', not 'value-judgment'. (Recall that we have already explained above how we understand the term 'value-judgment'.) It is also worth noting that our necessary and sufficient criterion explains why an algorithm producing hypothetical imperatives, such as the decision aid program discussed above, does not comprise essential value-judgments. If two or more software designers are asked to write a decision aid program it does not matter what ethical principles (or other value-judgments) the software designers themselves subscribe to. Since the aim of the program is to produce hypothetical imperatives, the algorithm itself can be entirely value-free.[7] It is certainly true that value-judgments are produced by a successful implementation of the program, but it is not true that two or more software designers accepting different value-judgments would ever have a rational reason to design the underlying algorithm differently.

We now come to a fundamental assumption in this paper, viz. that a software designer designing an algorithm is facing a decision process that is in many respects similar to that faced by people designing other technological artifacts. If true, this entails that value-judgments that are built into an algorithm will in many respects be similar to value judgments that feature in other design processes. For this reason it is helpful to elaborate a bit on the design processes from which algorithms and other technological artifacts are derived.

Let us first explain what we mean by 'design' by proposing a necessary condition we believe all such activities have to fulfill. (For present purposes, we can do without a sufficient condition.)

> DESIGN ACTIVITY: Something is a design activity only if it is an activity where a designer (or a team of designers) make decisions in order to reach a pre-defined goal.

We use the term 'variable' both for characterising what it means to 'make decisions' and for 'reaching a goal'. We assume that a variable stores exactly one piece of information. For instance, when designing a bicycle, the designer can decide that the front wheel shall have a diameter of 60 cm; this 'diameter of the front wheel' is a variable, and '60 cm' is the value stored in this variable. However, we must carefully distinguish the (four) different ways in which values can be stored in a variable.

First, the designer can *decide* that a variable shall have a particular value. For the rest of this paper, this settles our definitions of 'making a decision' as 'assigning a value to a variable', assuming that the decision-maker has the authority to assign the chosen value to the variable in question. For instance, no designer can decide that 'this bicycle will have a maximum velocity of 250 km/hour', since the maximum velocity of a bicycle is the result of a range of other decisions and empirical circumstances. Variables that may occur in a decision are called *decision variables* or *category-I* variables.

Second, a variable can take a value as part of the assessment of the degree to which a design-objective was met. For instance, suppose the bicycle we are designing should be as light as possible; then the value of the variable 'weight of the bicycle' helps to assess if we were successful. Such variables will be called *objective variables* or *category-II* variables. Notice that any category-II variable receives a value as the final consequence of assigning values to category-I variables, plus a number of (physical, economical, and social) mechanisms that causally propagate the category-I values through the designed artifact and its usage.

From the account sketched above, we see that category-II variables are (in a mathematical sense) a function of category-I variables: the design function. However, they are also a function of another class of variables. In order to see this, note that the weight of the bicycle depends on e.g. the density of the chosen material. If we

[7] It is worth noticing that some decision support systems may apply a reasoning algorithm that, although deterministic, is too complicated to be followed by human users (e.g., since it would take too long to be practical). In that case, the user cannot do anything else but follow up on the algorithm's suggestion *or* ignore the suggestion altogether. This renders the decision support algorithm value-laden.

choose carbon fiber (a possible value for the category-I variable 'material of the bicycle'), we have to accept that the density of carbon fiber occurs as an argument in our design function. Such variables cannot be controlled by the designer. We call them *context variables* or *category-III* variables. From the perspective of the designer, they are constants.

In all but trivial cases, the design function is immensely complicated. It is next to impossible to write down the values of category-II variables even if we know the values of category-I and category-III variables. We can regard it, however, as a composition of a large number of much simpler functions. The weight, for instance, is the sum of the weights of the two wheels plus the weight of the frame. The weight of one wheel is not a category-II variable (because the design is not necessarily better if this weight is less); it is also not a category-I variable (if we decide the diameter, the weight is no longer independent); finally it is not in category-III because it is not a constant. It is what we call an *auxiliary variable* or a *category-IV* variable.

This completes our general outline of the design process. What it amounts to is the establishment of a network of functional relationships between variables in categories I, II, III, and IV and the assignment of values to the category-I variables; the 'quality' of the design is assessed by inspecting the resulting values for the category-II variables. Consider Fig. 1.

The four categories characterised in Fig. 1 suggest an important link between ethics and the freedom of the designer: The choice of category-II variables may reflect, among other things, the designer's ethical view. For example, the designer might stipulate that the bicycle be constructed from recycled materials (reflecting the value of sustainability), and that it must be manufacturable in third-world countries (reflecting the value of geopolitical justice). In this respect we should observe that category-II variables come in two versions: *requirements*, that is: predicates that must be true (say, 'the bicycle shall be lighter than 15 kg'), and *desires* that relate to variables

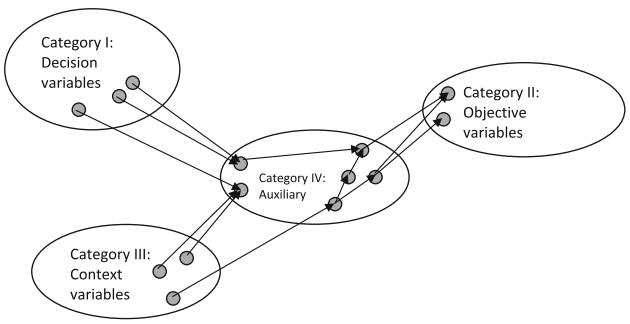


Fig. 1 Designing as a process of connecting 4 types of variables. *Arrows* indicate functional dependencies

subject to optimisation (say, 'the bicycle shall be as light as possible'). Among other things, the framework of four categories allows us to clarify trade-offs in the design, it helps us to pinpoint compromises, and it provides insight into possible design alternatives. We will use it here to decouple, on the one hand, the acceptance of an ethical position (such as the categorical imperative) from the implementation of the consequences of this position in a design (a hypothetical imperative): The first consists of deciding which ethical values should be represented; the second of representing these ethical values in terms of category-II variables.

A real example

In this section we discuss some examples of algorithms that are essentially value-laden. All our examples, one of which is real, are concerned with a particular kind of algorithm used in medical image technologies. Very briefly put, medical image technologies aim at representing human biological structures in computers in an accurate way, such as human organs and cells in blood samples, and thereby improve the diagnostic or therapeutic prospects of diseases affecting the biological structures in question. One of the many ethical issues raised by such algorithms is the risk of using algorithms that produce false positive and false negative results. By definition, a false positive result occurs whenever the algorithm triggers the system to count something (a cell, a symptom of a disease) in a digital image that is not actually there. A false negative occurs if an algorithm in a similar vein fails to identify a structure in the picture that is actually there.

For all practical means, it is virtually impossible to totally eliminate the risk of getting false positives and false negatives. However, if one is willing to accept a large number of false positive results one will typically get a smaller number of false negatives. In order to understand why, imagine you are asked by an eccentric millionaire to build a device that automatically counts the number of tigers in the jungle who passes through a certain spot. Naturally, if you build a device that simply counts everything yellow that passes through the jungle you will get a rather large number of false positive results, since e.g. bees, bananas, and yellow flowers will trigger the device. However, if on the other hand, you impose more conservative criteria on what is to count as a positive result and design the device such that only very large yellow items are detected, then you can expect the number of false negatives to rise (since e.g. new-born tigers babies will not be detected by the device).

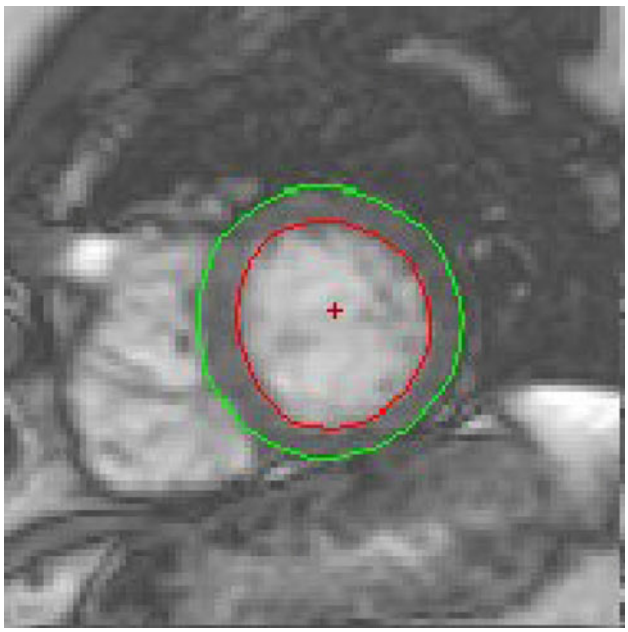
Clearly, software designers designing toy algorithms for detecting tigers, as well as real-life algorithms used in

medical image technologies, have to make a trade-off between minimising the number of false positives or the number of false negative results. This trade-off will inevitably be based on a value-judgment. There is simply no objective fact of the matter about whether it is more desirable to avoid a false positive or a false negative. Different users may have different preferences, and several conflicting preferences appear to be equally rational. It therefore seems hard to deny that an algorithm used for, say, counting the percentage of cells infected by a virus in relation to the number of non-infected cells will invariably contain a value-judgment about how many false positive results are tolerable in relation to the number of false negative ones. That said, both false positive and false negative results may give rise to severe negative effects for individual patients. Doctors base diagnostic as well as therapeutic decisions on what they come to believe about e.g. the number of infected cells in relation to the number of non-infected cells.

We can also illustrate our point about essentially value-laden algorithms by way of a real example. To start with, recall that in "Value-laden algorithms and the design process" we discussed some of the choices faced by designers by discussing the design of a material object (a bicycle). However, as pointed out in that section, the design process is roughly the same when designing immaterial artifacts such as algorithms. We can therefore use the general design framework introduced in "Value-laden algorithms and the design process" for illustrating how values for category-I variables propagate in the direction of category-II variables.

In the image to the right we see an MR-scan depicting a cross section of a human heart. For the purpose of diagnosing a variety of possible pathologies, it is necessary to accurately estimate the blood volume of the heart during various stages of a heart-beat cycle. The difference between blood and heart muscle tissue occurs as a difference in grey values in MR images. Estimating the blood volume starts with establishing which part of the image is colored lighter grey, and counting the number of pixels in this light grey area. This is called *segmentation* between light and dark, as depicted schematically in the diagram below.

One way of carrying out a segmentation is to use a numerical threshold, say T. This means that pixels with a lightness value above T are labeled 'light' and pixels with a lightness value below T are labeled 'dark'. The border between light and dark areas is, however, not sharp (Fig. 2).[8]



It is quite common to introduce thresholds in segmentation-like algorithms. Such thresholds serve as nice examples of category-I variables in the design process of the algorithm. There is no *a priori* correct value for such thresholds. There is no rigorous first principles from which they can be derived, since the noise in the image is an inevitable artifact of the MR measuring process, caused by numerous non-modeled sources. Software engineers typically choose a value that 'seems reasonable' for thresholds. In the example above we see how the choice for this threshold influences the estimated blood volume. This estimate will in turn affect further values, and eventually the diagnosis. So in borderline cases the diagnosis will (indirectly) depend on the value of the threshold T.

Of course, if the anatomical configuration is far from a borderline case (either 'very' healthy or 'very' pathological), there is obviously no problem. For the majority of pathological conditions, however, there is a continuum between 'healthy' and 'pathological'. Somewhere in this continuum there is a grey zone where the diagnostic outcome will critically depend on thresholds somewhere along the computational pipeline.[9]

[8] The algorithm outlined here is a deliberate simplification of actual, modern segmentation algorithms. In particular, the algorithms demonstrated in the cited website are much more sophisticated. For our argument, however, this is not relevant. Even advanced algorithms typically involve parameters, the values of which are to be chosen in order to decide, eventually, between 'normal' and 'pathological' cases.

[9] In principle, an algorithm can detect when it bases a decision on a 'near-borderline'-case. Sophisticated algorithms sometimes give an estimate of the reliability of their conclusions (for instance: long-term weather predictions, e.g. in the form of error-bandwidths). In this way, an algorithm can become less value-laden, since it moves part of the responsibility in the direction of the (human) user.

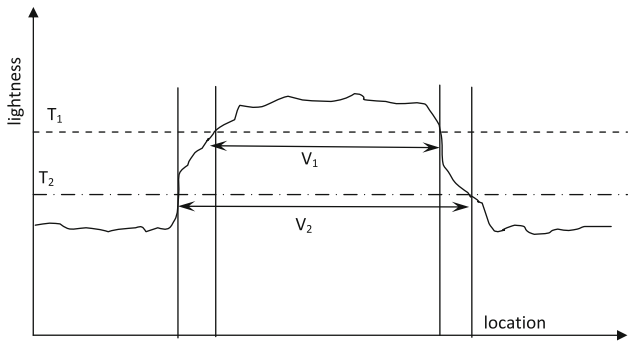


Fig. 2 Segmentation in a noisy image. For a larger threshold value, T1, the estimated area V1 of the 'blood' segment will be lower than the estimated area V2 which is found with a lower threshold T2. It is not a priori clear, however, which of the two threshold values is the 'correct' one. The software engineer chooses a threshold without real argument—thereby biasing the outcome of the algorithm when it is used on patient data. This may statistically influence the change of false positive diagnostic errors in favour of false negatives, or vice versa

Now, given that we take the computational output of a medical diagnosis support system to be a designed artifact, we can ask ourselves what the category-II variables are in this example. Apart from trivial ones, such as patient safety during clinical measurement, and operational comfort for the medical specialist, we may want the quality of the (automated) diagnosis to be 'good'. But the notion of goodness must of course be rendered more precise, and this is exactly where our observations concerning false positives and false negatives come in. In borderline cases the bias towards false positives or false negatives may to a large extent be determined by the values of thresholds or other category-I variables.

At this point it could be objected that the best way forward is to design the algorithm such that it totally eliminates all false positive and false negative results. However, a closer analysis of the risks for getting false positives and false negatives in medical images shows that this is not likely to be a feasible option. First, all algorithms that are applied to a large set of measurements will be sensitive to stochastic effects in the data set, and this will automatically yield false positive and negative effects. Second, it should be noted that the images are in most cases numerical (re-)constructions of large amounts of physical measurements. This point has important consequences: Nowadays the quality of such computer generated images is getting close to being photo realistic. That is, the data is presented (rendered) as if it is an actual photo of some 3-D internal organ or tissue structure, perhaps even isolated from its environment. Such rendering or isolation, however, requires complicated algorithms that include segmentation (as mentioned above), as well as much other subtle image processing. So rather than being objective photographs, the 3D medical images that specialists look at (and base their diagnosis on) are the result of an elaborate algorithmic process, depending on arbitrary thresholds in a difficult-to-predict manner. Now, medical specialists, like any other humans, are accustomed to interpret plausible 3-D images as accurate projections of corresponding 3-D objects. It is very difficult *not* to interpret a realistically looking 3-D image as a trustworthy projection of a 3-D object. This introduces the risk that one will forget that in order to generate these 3-D images, a number of decisions about thresholds had to be taken. Had some decisions been taken differently, the image could have looked (very) different—and, as we argue, there is no a priori way to give the correct value for such thresholds.

Understanding the relations between what is seen in the image and what really exists in a patient's body (or in the microscopic slice of tissue being examined) therefore requires a thorough understanding of the various algorithmic steps that are applied to the physical data in their transformation towards a collection of colored pixels. Such transformations include a choice of filters (involving variables with values that may significantly affect the eventual 'looks' of the image – and therefore the conspicuousness of certain types of anomalies) and perhaps geometrical algorithms (such as contouring, segmentation, tracking, etcetera). These filters and algorithms have to be chosen with some goals in mind – but these goals cannot typically, for practical reasons, be formulated in terms of minimising the risk of acquiring false negatives.

The third and last reason why it is not feasible to design algorithms that avoid all errors can be formulated as follows. Many (semi)automated diagnostic tools have been developed in recent years, and these diagnostic tools typically incorporate implied hypotheses of expected pathologies. For instance: a probe to identify a potential aneurism, stenosis or tumor can only do so because it *expects* certain characteristic features of, say the shape or volume of such anomalies. An experienced medical doctor realises when she sees something 'unexpected'; a software program usually cannot. Again, these diagnostic tools may not have been developed with the goal of minimising the percentage of false negatives in mind. Finally, there is one further problem. Software for complicated tasks such as medical diagnosis is often immensely complicated. Much if it consists of components that may have been developed earlier for 'general purposes'; these components are re-used in order to make the software production process economically feasible. Segmentation algorithms are an example of such components. Components are preferably treated as black boxes: based on their formal, functional specifications, their behavior can be assumed to be 'correct'. The ethical position, however, that was adopted during the construction of such a component when either choosing a more conservative or more liberal threshold, is

typically not part of their formal specification.[10] That means that the same segmentation algorithm, applied in two different systems, will behave equally with respect to its formal, functional requirements, but at the same time it may behave oppositely with respect to its tendency to produce false positive or false negative judgments.

The total complexity of highly interrelated physical techniques, algorithms, heuristics and visualisation methods is immense, and there is little hope to optimise the entire, integrated chain in terms of minimising the chance of false negatives. What may be feasible, however, it to take this chain apart in a systematic way, and to analyse the various stages as if they were (to some extent) independent. This route may be facilitated to use a framework such as the 8-layers model developed by one of the authors to argue about the information contents in images. This model separates this information into coherent chunks (e.g., the shape-related information is separated from the texture-related information, which is separated from the 3-D surface related information, etcetera.)

The precautionary principle and ethical theories

In this section we discuss two different ways of addressing ethical issues raised by the risk of obtaining false positive and false negative results when using medical image algorithms. In the next section we shall identify and discuss some ethical issues that apply to algorithm design in general.

Briefly put, we propose that a possible way of managing false positive and false negative results obtained from medical image algorithms is to adopt an epistemic interpretation of the precautionary principle.[11] The second option, which we will also explore in depth, is to approach this choice as an ethical judgment to be determined by an ethical theory.

Let us first consider the precautionary-oriented, epistemic strategy. The precautionary principle was originally invoked by policy makers for addressing environmental issues, such as global warming, toxic waste disposal, and marine pollution. In recent years it has also been suggested that the precautionary principle may also be applied to medical issues. David B. Resnik argues that "properly understood, the [Precautionary Principle] can provide physicians and patients with a useful approach to medical decision making." The precautionary principle can, however, be interpreted in many different ways. According to the epistemic (belief-guiding) interpretation outlined in Peterson (2007), the precautionary principle should be characterised in terms of what it urges us to believe. To put it more precisely, the epistemic version of the precautionary principle holds that it is better to get a false positive rather than a false negative result, contrary to what is commonly accepted in science, since this will prevent doctors and patients from falsely believing that someone who is in fact ill is healthy.

Scientists generally agree that it is more important to avoid false positives than false negatives. This is because scientific knowledge tends to be cumulative: We add new beliefs about what the world is like to a set of already existing beliefs; and the justification for the new beliefs typically depends in more or less intricate ways on the old beliefs. Therefore, if we start to accept too many false positive beliefs we may end up in a situation in which future research is directed in the wrong direction by our false positives. A more conservative approach, in which false negatives are preferred over false positives, is more likely to be successful in the long run, since it makes it more likely that we will not base new beliefs on old but false ones.

However, when discussing medical image algorithms it is far from clear that we should adopt the same set of (epistemic) values that guide, or at least ought to guide, scientific research.[12] The reason for this is that the consequences of falsely believing something to be safe when it is not might be disastrous. If the algorithm is designed such that doctors come to believe that patients who are actually diseased are not, then the doctors may indirectly cause indirect harm to patients by failing to treat them. We therefore conclude that when addressing ethical aspects of medical image algorithms it is far from clear that medical decision makers should prefer algorithms that make them believe what is most likely to be true. On the contrary, a strong case can be made that medical image algorithms should be designed such that they are more likely to produce false positive rather than false negative results. That said, it is of course essential to make sure that an increased number of false positives does not lead to too many unnecessary and potentially dangerous operations. The computer image is just a tool. The final responsibility for

[10] By 'ethical position' we mean the decision at which level the threshold should lie. We will deal with this in detail in the next section.

[11] This section is partly based on some ideas first presented in Peterson (2007).

[12] To be precise, we should distinguish between on the one hand, medical imaging algorithms used exclusively in interaction with a medical specialist, and on the other hand, algorithms used to screen large volumes of data to seek for pathologies. In the latter type of cases the data volume is too large for (human) medical staff to handle in given clinical contexts (consider e.g. yearly screenings of ten thousands of women for early signs of breast cancer). In such cases, the risk for statistically significant biases in the results (that are ethically value-laden) is much larger than in the first case.

deciding whether a surgical intervention is appropriate has to be taken by the doctor.

It is important to bear in mind that our conclusion about the importance of avoiding false negatives in medical image algorithms is a value-judgment, not a factual one. No observations or other purely empirical methods can be appealed to for backing up this conclusion. This is part of the explanation of why medical image algorithms are essentially value-laden. However, as with nearly all value-judgments, good reasons can also be given against this value-judgment. For example, it could be claimed that the software designer designing the algorithm ought to perform a detailed cost-benefit analysis of the pros and cons of accepting different trade-off rates between false positives and false negatives. This is an approach that, broadly speaking, tallies well with traditional consequentialist intuitions. It may turn out, for instance that the total utility of accepting some false negatives but avoiding a very large number of false positives may actually be optimal, because of various empirical circumstances. It is beyond the scope of this paper to take a definitive stand on whether the consequentialist view outlined above is at least as plausible (or even more plausible) than the precautionary approach. Here we just wish to highlight that the choice of a trade-off rate between false positives and false negatives is a genuine value-judgment, and that this makes medical image algorithms essentially value-laden.

Let us now consider an alternative, slightly more theoretical approach to the choice between false negatives and false positives. The ethical assumption underlying this approach is in short the following: A doctor's decision about whether avoiding false negatives is better than avoiding false positives should depend in part on the answer to the question of how dangerous the suspected disease is for the patient. The more dangerous it is, the more important it will be to detect it, and the more acceptable it is to take the risk of mistakenly identifying it in a patient (i.e. to produce a false positive). The same holds, secondly, for contagious diseases that could jeopardise others. Third, a rising number of false negatives is less acceptable if the faulty diagnosis is likely to bring about drastic side-effects such as e.g. a dramatically diminished quality of life after diagnosis, or even the triggering of suicidal tendencies in those who are mistakenly informed of a positive result in the testing of a disease. For instance, the mass screening for breast-cancer in women has recently raised severe criticism due to the fact that it bears a high risk of false-positives. Being wrongly diagnosed with breast cancer is usually extremely disturbing for women and their families.[13]

[13] See e.g. Gigerenzer et al. (2009) about these problems of screening and their statistical implications.

The choice for a certain threshold in an algorithm is a decision that is a judgment about which there is, or at least could be, a controversy between advocates of the major theories of normative ethics. This supports our claim that algorithms manifest or reflect certain ethical judgments. This point can be illustrated as a choice between deontological and consequentialist or utilitarian theories of normative ethics.

Deontologically minded users of the software are most likely to focus on the physical and mental integrity of the individual patient. Therefore, in a Kantian vein, medical doctors will most probably opt for the implementation of some more liberal algorithm when it comes to computerised image analysis of severe diseases, aiming at more false positives and less false negatives. In such a Kantian vein, it is pre-eminent to protect the individual patient, i.e. to avoid doing harm to him or her. Such harm would be most likely to be done if a severe disease remains undetected and therefore untreated. For this reason, the deontologist is willing to put up with problematic consequences of more false positives that are brought about by her choice.

In contrast, a scientist who tries to gather statistical data about a population of patients may wish to use the same software, but she will typically prefer more strictly or conservatively designed algorithms. This will bring about results that are more in line with consequentialist or utilitarian decision-making: What is important in this perspective is that the body of scientific data as a whole remains valid and intact, because this will typically bring about better consequences in the long run, although some individual patients may have to suffer along the road. This is in accordance with the consequentialist or utilitarian view that overall well-being in society should be our focus of attention, rather than the rights or needs of single individuals. Therefore, on this view, aiming at scientific insights that benefit the greatest number of people is the highest priority. This means that within the consequentialist approach, scientific results should be free from false positives as far as possible. It is therefore acceptable that a relatively large amount of false negatives will arise.

It is beyond the scope of this paper to take a stand on the controversy over deontological and consequentialist normative theories. However, it is important to note that the choice of a threshold value for a certain algorithm goes hand in hand with some normative background assumptions specified by the software designer. Problems arise if it is unforeseeable whether the background assumptions of the designer will be in accordance with those of future users (i.e. a doctor or a scientist). If the ethical assumptions made by the software designer remain implicit and differ from those preferred by the user, we face a rather severe problem: Without knowing it, the user who bases her

decisions on the output of the software will base those normative choices on ethical assumptions that are in conflict with the implicit ethical assumptions made by the software designer.

Should ethical values be user-defined?

Our general view is that software designers should, as far as possible, leave ethical decisions to users (say, with an external switch to choose between 'preference for false positives' or 'preference for false negatives'). Moreover, when this is not possible the ethical assumptions in the algorithm should at least be transparent and easy to identify by users. In the case of medical image algorithms the users amount to medical doctors and scientists. For the reasons outlined in "The precautionary principle and ethical theories", physicians who use a software will typically work on the assumption that it is better to get a false positive than a false negative. But when scientists use exactly the same software they are likely to have the opposite ethical view. Such users will typically avoid false positives in the first instance. These divergent ethical judgments could, of course, easily be implemented into the software by including some code that allows the user to specify her preference between false positive and negatives.

We are aware that such a leave-it-to-the-user approach may sometimes be impossible to implement, as indicated above. There are cases in which it may just not be feasible to directly ask users to express the required kind of ethical judgments in numerical or quasi-numerical terms. Having to take a stand on fundamental ethical issues before using a software may be too difficult, and it may also be too impractical to ask users to make the relevant ethical choices. In such cases the underlying ethical assumptions should at least be made transparent. However, in what follows we shall set such cases aside and focus on cases in which it may actually be feasible to eliminate ethical judgments as far as possible from the algorithms underlying the software.

At the heart of the matter is the fact that it is not just a practical decision that must be made when designing an algorithm, but a genuinely ethical one. To opt for either a consequentialist or utilitarian or a deontological attitude towards e.g. the choice between false positives and false negatives amounts to a value judgment that brings about ethically relevant implications for the patient's health, for those who are in touch with her, and for the progress of science as a whole. As explained above, we therefore advocate that the software designer should design the algorithm such that it remains flexible and applicable to the requirements of different ethical settings. To be more precise, we propose that algorithms as far as possible remain open to the user's ethical preferences. The design of the algorithm must allow the user to choose the circumstances in which she situates herself. We therefore think that it is not enough that the designer makes his or her assumptions transparent by letting the user know what the ethical assumptions in the design stance were. Rather, it is necessary that the designer leaves it to the user to specify what ethical parameters to choose.

Having said this, we are of course aware that we now go beyond what is usually required in the literature on the ethics of image analysis. One of the few comments on the ethics of image analysis stems from Gert-Jan Lokhorst who was interviewed by van Strien (2008). He suggests that software has to be trained on image materials that resemble the ones it will have to analyse in the clinic, because the software will recognise and identify only those matters on which it was explicitly trained before. Moreover, Lokhorst suggests that doctors have to undergo proper training before they use the software. This will ensure that users become aware of which sorts of image analysis samples the system is familiar with, and this in turn ensures that users will be able to properly assess the competences of the software. Lokhorst's final suggestion is that users and other relevant parties should be provided with some basic knowledge about the situations in which the algorithms were developed by the designer, because (i) this will enable them to decide in which clinical situations the program should be applied and in which not, and (ii) it must also be made clear in advance who is responsible for which step in the process of image analysis.

Lokhorst thus implicitly agrees with us that it is essential to make the software designer's ethical choices transparent to the user. However, our way of formulating this ethical requirement is more far-reaching than Lokhorst's. As explained above, we maintain that the software designer must leave the responsibility for defining the default state of the software to the user herself. This is required for avoiding possible discrepancies between the ethical background assumptions of the software designer and users.

The example of the stenosis in "A real example" shows that there is no a priori "correct" threshold value. However, the very first choice made by the software designer will influence all further steps of diagnosis. In borderline-cases, those choices may even affect the treatment suggested to the patient by the doctor. This is disturbing, especially in face of the fact that the software designer may lack the proper medical background of a fully trained physician.

According to our argument for implementing user-defined ethical values, the doctor treating the individual patient should be enabled to make the relevant choice herself, as far as possible. If the patient is facing a severe condition, the threshold value should perhaps be set more

liberally, to make sure that at any sign of deterioration, treatment is initiated. However, if the health condition of the patient is estimated as overall good, the threshold value should be set more conservatively to avoid "false alarms" that could lead to surgical interventions causing unnecessary harm to the patient.

Conclusion

We conclude that a strong case can be made for the claim that some algorithms are essentially value-laden. Some algorithms, such as those used for classifying cells as diseased or non-diseased, forces the designer of the algorithm to take a stand on controversial ethical issues, e.g. whether it is more desirable to prefer false positive errors over false negative ones. This is a controversial ethical issue, on which there is a lot of disagreement among ethicists. We propose that designers of algorithms should, as far as possible, leave ethical issues to users, and when this is not possible, the ethical assumptions in the algorithm should at least be transparent and easy to identify by users.

Acknowledgment The authors wish to thank Robin van der Sligte for extremely helpful discussions, and Sven Diekmann and Rosemary Lowry for helpful comments on earlier drafts.

References

Allen, C., Wallach, W., & Smit, I. (2006). Why machine ethics? *IEEE Intelligent Systems, 21*, 12–17.

Arkin, R. C. (2009). *Accountable autonomous agents: The next level.* Position paper for the DARPA Complete Intelligence Workshop, Feb. 2009.

Gigerenzer, G., Mata, J., & Frank, R. (2009). Public knowledge of benefits of breast and prostate cancer screening in Europe. *Journal of the National Cancer Institute*. doi:10.1093/jnci/djp1237.

Peterson, M. (2007). Should the precautionary principle guide our actions or our beliefs? *Journal of Medical Ethics, 33*(1), 5–10.

Resnik, D. B. (2004). The precautionary principle and medical decision making. *Journal of Medicine and Philosophy, 29*(3), 281–299.

Sipser, M. (1997). *Introduction to the theory of computation*, PWS Publishing Company.

Turilli, M. (2007). Ethical protocols design. *Ethics and Information Technology, 9*, 49–62.

van Strien, W. (2008). Opvangen zwakheden maakt beeldtechnieken waardevoller (interview met G.J.C. Lokhorst). *Ethiek, Onderzoek en Bestuur*, pp. 20–25. March 2008.