# What the Hack Is Wrong with Software Doping?

Kevin Baum(✉)

Saarland University, Saarbrücken, Germany
k.baum@uni-saarland.de

**Abstract.** Today we often deal with hybrid products, i.e. physical devices containing embedded software. Sometimes, e.g., in the VW emission scandal, such hybrid systems aims rather at the fulfillment of interests of the manufacturers than at those of the customers. This often happens hidden from and unbeknown to the owners or users of these devices and especially unbeknown to supervisory authorities. While examples of such *software doping* can be easily found, the phenomenon itself isn't well understood yet. Not only do we lack a proper definition of the term "software doping", it is also the moral status of software doping that seems vague and unclear. In this paper, I try, in the tradition of computer ethics, to first understand what software doping is and then to examine its moral status. I argue that software doping is at least pro tanto morally wrong. I locate problematic features of software doping that are in conflict with moral rights that come with device ownership. Furthermore, I argue for the stronger claim that, in general, software doping also is morally wrong all things considered – at least from the point of view of some normative theories. Explicitly, the VW emission scandal is adduced as a significant specimen of software doping that unquestionably is morally wrong all things considered. Finally, I conclude that we ought to develop software doping detection if only for moral reasons and point towards the implications my work might have for the development of future software doping detection methods.

## 1 Introduction

How many computers do you own? Many people cannot answer this question correctly these days – because we are typically not aware of the fact that we deal more and more often with mixtures of physical devices and software. Those *hybrid* products are, strictly speaking, computers – or, at least, contain one or more of them. Such hybrid devices come in many shapes and sizes, ranging from obvious examples like printers and smartphones to cars and toys. Hybrid devices raise a number of important legal as well as philosophical questions.

For instance, recently the case of the VW emission scandal hit the news: The German car manufacturer used an embedded software for detecting emission

K. Baum want to thank Holger Hermanns, Stephan Schweitzer, Stephan Padel, Deborah Hirth, and David McCann for their helpful comments on the ideas presented here.


T. Margaria and B. Steffen (Eds.): ISoLA 2016, Part II, LNCS 9953, pp. 633–647, 2016.
DOI: 10.1007/978-3-319-47169-3_49

testing and tweaking the engine's behavior to fit the legal emission requirements. We can learn a lesson from this case: Not every software embedded in a device is essential to the purpose and proper functionality of it. Rather, it may modify the device in ways that solely aim at the fulfillment of the manufacturers interests, and – by doing so – ignore the interests of the users and owners. Typically, such software works in the background, hidden from and unbeknown to the users and owners of the devices; it is often even clandestinely added to deceive supervisory authorities. This so-called *software doping* allows VW to pass nitrogen oxide emission tests by cheating, manufacturers of printers to reject third-party cartridges, and point-and-shoot camera manufacturers to hide a bunch of features from their users in cheaper models, making them available only in more expensive models. Examples of software doping can be easily found, it seems. However, the phenomenon itself isn't well understood yet. Although the term "software doping" is made-up and rather a technical term, it has yet to be properly defined. While we have an intuitive grasp, conceptual work is pending. But even putting these conceptual worries aside, questions concerning the moral status of the practice of software doping are in need of answers.

This paper aims at taking a step towards solving both of these issues, putting to use methods of moral philosophy, especially from the fields of normative and computer ethics. First, I suggest a definition of "software doping" in terms of input-output modification of devices aiming at the benefits of the manufacturers at costs of certain owner interests in Sect. 2. Then, in Sect. 3, I first argue that software doping is necessarily, by definition, pro tanto wrong, followed by the stronger claim that, in general, software doping also morally impermissible in an overall sense – at least from the point of view of some normative theories. This will be made clear by exploiting the VW emission scandal as a vivid example. Finally, I will distill some lessons to learn from the result and sum up my findings in Sect. 4.

## 2 What Is Software Doping?

Analytic philosophy is often concerned with providing conceptual clarity – not as a task for its own sake, but because it is taken to be a necessary precondition for achieving deeper knowledge about our understanding of the world. Some observations turn out to be fundamental for such philosophical endeavors: Terms and concepts are often understood and used without us being able to clearly define them. Sometimes, we can even *apply* conditions of correct and incorrect usage of terms *without being able to spell out* those conditions immediately – we may be able to correctly say that in a certain context the use of some word is clearly incorrect without being able to say why this is so and how we know.

Note, that this is not a grammatical point. Rather, it is the idea that an intuitive understanding of a term is enough to correctly apply the term without being able to formulate some criterion: We can apply every-day terms like "love" or technical terms like "internet" correctly/adequately (and also can identify incorrect/inadequate usage) without being able to spell out proper definitions of

the terms or explicit rules of correct usage.[1] However, in order to find answers to deep questions regarding the nature of certain concepts (denoted by some terms), we need at least something like characterization, operationalizations, or even better real definition of the corresponding terms: We need to do *conceptual work*. This methodology can be applied to all kinds of terms, even to artificial and technical terms like "software doping". This results in a better understanding of the related concepts.

While conceptual work is central to analytic philosophy in general, it is also perfectly in line with what James Moor identified as the classical nature of computer ethics in his very influential paper "What is Computer Ethics?":

> A typical problem in computer ethics arises because there is a policy vacuum about how computer technology should be used. [...] A difficulty is that along with a policy vacuum there is often a conceptual vacuum. Although a problem in computer ethics may seem clear initially, a little reflection reveals a conceptual muddle. What is needed in such cases is an analysis that provides a coherent conceptual framework within which to formulate a policy for action.[2]

Accordingly, this section deals with conceptual work, striving to identify and overcome a conceptual muddle. I try to explicate different aspects from the intuitive notion of software doping and then test these particular aspects against some example cases in order to sharpen our intuitive notion and to arrive at a full-blown and – most importantly – useful concept.

### 2.1 Defining "Software Doping"

Every conceptual work needs a starting point. Fortunately, the track organizers offer a useful characterization of "software doping" in intuitive terms: "Embedded software might provide features and functionalities that are not in the interest of the device's user or even of an entire society." And then:

> Embedded software doping is what enables inkjet printers to reject third-party cartridges, and it enables cars to secretly pollute our environment. In general terms, embedded software doping locks the users out of the products they own.

The term's meaning seems clear on an intuitive level, but the provided definition seems extensionally inadequate. Besides, these passages don't seem to tell the full story: for instance, providing features that are not in the interests of the device's

[1] Especially, such capabilities are not sufficient for philosophical work concerned with more theory-laden (philosophical) terms. For instance, while many of us use the terms "responsibility" or "knowledge" frequently and without any further thoughts about correctness conditions or definitions, a lot of philosophical work is concerned with how these terms can be defined or at least characterized properly.

[2] [8], p. 266.

users is neither equivalent nor obviously or immediately linked to being locked out of a product one owns. So the questions remains what software doping is exactly.

We can start answering the question by having a look at term's origins: The term "doping" is obviously borrowed from the field of sports. This already suggests two features: First of all, the device's 'behavior' is modified in some sense – and in the case of *software* doping this is achieved by the use of software (we are not concerned with doped software). Secondly, this is done intentionally with the aim to secure an (unfair and improper) advantage for the manufacturer. Accordingly, it is typically done secretly. To stretch the analogy to its limits, think of the manufacturers as the countries, trainers or teams of doped athletes and of the owners and users as fans. Now, let us take a look at the different aspects, I have extracted so far and clarify why and whether they are necessary for adequate definition of software doping.

**A Device's Behavior.** Taking a black-box perspective on the device allows us to shift the focus on the 'behavior' of a device is and enables us to express the influences of software doping: it may modify the inputs accepted by the device (as in the printer example) or the outputs that are generated, given specific inputs and a context (as in the VW emission case).

**Intentionality and Aiming at Advantages.** It is central for the concept of software doping that manufacturers add the software intentionally and with the aim to take (unfair) advantage of it, as an example makes clear: Recently, a U.S. federal magistrate judge ordered Apple to backdoor the iPhone that was used by one of the perpetrators of the San Bernardino shootings in December 2015.[3] The legal case was closed after the FBI was able to access data without the help of Apple. Now, assume – contrary to the current facts – that all manufacturers of smart phones are enforced by law to implement backdoors into every smart phone. These cases don't seem to be software doping in the relevant sense. Therefore, an appropriate account of software doping should consider the intention of the manufacturers in order to exclude such cases.

**Secretly?** Furthermore, it has to be stressed that according to the above characterization software doping isn't *necessarily* done in secret. Typically, it is – especially when it is done illegally. But the case of the voluntary self-restraint of most German car companies to a top speed of about 155 mph isn't something unknown to the public. Nevertheless, it seems to be at least a borderline case of software doping[4], since it was originally done in the interest of the manufacturer: First and foremost because they assumed future problems with safe tires for higher speeds, but second to forestall possible interventions by legislature (in fact, the self-restraint is crumbling the last years).

**Against the Owner's Interests.** This is, however, still not the full story. Imagine that a manufacturer adds a feature to its devices with the sole

[3] See the original order on documentcloud, [10], the statement of Apple, [4], and the statement of the Electronic Frontier Foundation, [9].

[4] Only if it is done by using software, which is not the case in general, but is becoming more common.

intention to benefit from it relative to competing companies. This doesn't mean that the feature is not as well in the interest of the users or owners of the devices, even if this wasn't originally thought of. For example, in the early ages of digital media, companies could have decided to completely ignore Digital Rights Management *just because* it was complicated to implement. This clearly would have been in the interest of many if not all owners of such devices. Thus, one should add that software doping modifies the affected device's functionality against the legitimate and justified interests of the owners – something I will explain in detail in a second.

All in all, this leads to the following definition:

**Definition 1.** *Embedding software*[5] *S in a device D is* software doping *if and only if S was put there intentionally and modifies the device's behavior (the inputs accepted or the outputs generated by D) in a way that isn't in line with the justified interests of the owner/user, but instead serves the interests of the manufacturer.*

However, whether this definition is satisfactory highly depends on how "justified interests" is spelled out in detail – the endeavor of the next section.

### 2.2 Owner's Justified Interests

What distinguishes justified interests of owners from interests of owners that a manufacturer is permitted to ignore or dismiss isn't a trivial question. In the following, I first give a negative answer, i.e., what kind of interests are paradigmatic cases of unjustified interests. Then, in a second step, I provide a positive answer to the question by introducing a kind of interest that clearly qualifies as justified.

Types of interest that are not justified interests of owners are easy to identify. Although one might be inclined to believe that every interest regarding the use of one's property is justified, this turns out to be an overstatement. Some interests of owners may well be neglected by embedded software without intuitively rendering these cases instances of software doping – even if it is done intentionally by the manufacturers and in their own interests:

**Interests in Unrealistic Purposes.** Assume you buy some product that you would like to use for a specific purpose, but that purpose is beyond the scope of what the device could do. If you have an interest in using it for such a purpose, then these interests are unjustified, because you don't have any entitlement to expect the unrealistic.

For example, think of a customer who wants to turn the volume up on his amplifier, but the software blocks at a certain level – a level, though, that

[5] My use of the term "software" here should apply also to components and/or subroutines of whole programs. Programs that might well be necessary for the hybrid device's overall functionality.

corresponds to what the hardware can maximally achieve (and this is not due to some suboptimal wiring or so). Since the interest simply cannot be fulfilled, it is thus unjustified.

**Interests against Essential Functionality.** Assume you buy a product and it does what it is supposed to do. However, there is a piece of software that somehow conflicts with your interests in a deep sense – i.e. you don't just dislike a contingent feature like some color or sound choice, but something that is essential to its functionality. Without the software, a basic feature of the device wouldn't work anymore, but you nevertheless want it that way. As long as you were free to buy the product and you knew (or to be more precise: an *idealized* version of a consumer in the same circumstances easily could have known[6]) what you bought, this is not a justified interest.

Take as an example a new navigation device: After using it for the first time, you notice that you hate the fact that a computer is giving you directions. You don't just dislike the implemented voices, but the very idea of being guided by a computer.

**Interests in Illegal Purposes.** Assume you buy a product that in principle could be used for illegal ends, but you are prevented from using it for that purpose by embedded software and this limitation is even enforced by (morally acceptable[7]) laws. If you have interests in using it for this illegal purpose, then these interests are illegitimate and thus count as unjustified. Assume you want to play a pirated copy of a movie on your standard player device, but your player doesn't accept the copied Blu-ray disc due to missing DRM information. This might not count as a clear case of a justified interest of the owner. If we further assume that the manufacturer of the player was under a legal obligation to implement that DRM checking routine, it seems to be clearly no case of a justified interest.

This list may well be not exhaustive. Nevertheless, the idea behind what is *not meant* by "justified interests" should be clear now.

However, at least one positive example is needed to clarify what *is meant* by "justified interests of owners". Looking at examples of unjustified interests, we can summarize that by talking about justified interests of owners in their property we mean interests in using property in ways at least compatible with the originally intended way. Nevertheless, there might well be justified interests that go further, beyond the originally intended use and proper functionalities of the devices owned. To that end, I derive only a sufficient condition from this first aspect here. For this, I first need to try to make sense of the idea of the proper functionality of devices in terms of a black-box view on them.

[6] For instance, the consumer's lack of intellectual capacities should not suffice to render the interests justified.

[7] To be precise, this qualification is needed: If the laws are highly morally inadequate, my interests in using my device against the law may well be legitimate. However, to make things easier, for the rest of the paper, I make the pragmatic assumption that the laws relevant in cases of software doping of interest are morally acceptable in the relevant sense.

**Definition 2.** *Let $D$ be a device. $D$'s* proper functionality *can be expressed by a set of input types $\tau_{\mathcal{I}_D}$, a set of outputs $\mathcal{O}_D$, and a function $f_D : \mathcal{I}_D \times \mathcal{C} \rightarrow \mathcal{O}_D$ where $\mathcal{I}_D$ is the set of possible inputs of the types in $\tau_{\mathcal{I}_D}$ and $\mathcal{C}$ stands for the set of contexts for the use of $D$.*[8] *It must hold that, in the process of acquisition, these three components were* knowable in a transparent way *to the future owner or at least they were* reasonable to assume *for her.*

Here we may assume something to be *knowable in a transparent way* if it is advertised in that way, it was told by the authorized salesman to be that way, or it is part of all the principally openly accessible information (handbooks, necessary features to be compliant with the law, et cetera). Similarly, we may understand something to be *reasonable to assume*, if there is something, I obviously can do with the product, or there is a standard or default that normally is in place for that type of product – like, e.g., laws. In the case of the VW emission scandal, it was a necessary feature of the diesel cars sold to be in compliance with the laws to respect certain emission thresholds. Thus, consumers justifiably assumed that the cars they bought emit no more than the allowed amount of $NO_x$. Regarding the input side, note that constraints may be in place by default, determining admissible types in $\tau_{\mathcal{I}_D}$ – mostly, they are legal issues. E.g., it is reasonable to assume that if my car drives on some specific *type* of gas, the *brand* won't be a distinctive feature: you can buy gas of a certain type at a filling station of your free choice. So it is – or ought to be – the case with ink: if my printer can achieve the same quality in print with ink having certain chemical and physical properties, delivered in specific cartridges of a fitting shape, then, again, the brand should not matter.

Definition 2 allows us to arrive at a clearer view on justified interests:

**Condition 1.** *An owner has* justified interests *regarding her property $D$, if her interests are compatible with the proper functionality of $D$, i.e., it is possible that those interests are fulfilled while the proper functionality is in place.*

Note that none of the unjustified interests from the last subsection fulfill this sufficient condition: In the mentioned examples it cannot be the case that the sketched interests are fulfilled in light of what is being transparently knowable or reasonable to assume by the owners about the $\tau_{\mathcal{I}_D}$, $\mathcal{O}_D$, and $f_D$ as components of the proper functionality of the products in question. Hence, the interests in those examples are not compatible with the proper functionalities of the products.

Condition 1 may seem unspectacular in the sense that it only allows for a very limited number of kinds of interests or a pretty 'boring' sort. But on the contrary, it also allows for more special sorts of interests. For instance, take the following kind of interests:

**Interests in Originally Unintended Purposes.** Assume you buy some product, which, in principle, could be used for an originally unintended purpose.

[8] This is, admittedly, a very simplistic model. But, I am confident it suffices to make clear the general idea while being easily adoptable to more sophisticated models.

If you have interests in using your property for such unintended purposes, but you are prevented from that by some embedded software, then your interests should not necessarily count as unjustified – by definition –, because your interests still may be in line with the proper functionality of the device. To see this, forget for a moment about hybrid devices and think instead of 'good old' property. For instance, there is a whole community that is concerned with so-called "life hacks", i.e., the creative usage of things in an originally unintended way in order to make life easier or to come up with more efficient solutions to everyday problems. Indeed, nobody would declare your use of some binder clips as, say, cable holders as an expression of an unjustified interest in using them. This use also would be perfectly in line with their proper functionality: After all, they are sold as something you can use to clip something to something else. Even if the original intention of the manufacturer or seller may have been that you clip a bunch of papers together, this just means that they didn't had all plausible 'inputs' in mind originally. However, remember that the frustration of such (justified) interests does not suffice to render the candidate case an instance of software doping since for this it must additionally hold that these interests are frustrated to serve the interests of the manufacturer instead.[9]

Now that we have a useful and intelligible characterization at hand, we are able to analyze the moral status of software doping.

## 3 The Wrongness of Software Doping

So far we thinned out the conceptual fog around the concept of software doping by giving a sufficient condition for software doping. This enables us to analyze the moral properties of the concept. In a first step, I explain what is pro tanto morally wrong with software doping.[10] Then, I work out the role of normative background assumptions when it comes to the question whether, as default, software doping should be considered morally wrong all things considered. Finally, I argue for such default view motivated by the VW emission scandal.

[9] Nevertheless, there might be something morally wrong with the frustration of the owner's justified interests in such cases. This even might be the case for very similar reasons, I invoke in a moment to show the wrongness of software doping.

[10] The terms "pro tanto" and "all things considered" are common technical terms in analytic philosophy. For instance, there is a difference between *pro tanto wrongness/rightness* and *all things considered wrongness/rightness* or *overall wrongness/rightness*. Or so it is often assumed in ethics, at least by many mainstream views. A pro tanto wrongful action may well be the right thing to do all things considered. For example, think of harming someone (a pro tanto wrong thing to do) in order to prevent a lot of much more intense harm to many others. Or think of lying to someone (pro tanto the wrong thing to do) in order to save another's live. In both cases people have strong intuitions that a pro tanto morally wrong thing is the overall right thing to do.

### 3.1 Why Software Doping Is Pro Tanto Wrong

With ownership comes property. According to modern theories of property, ownership and property come with – or even define or constitute – certain rights.[11] The two least controversial rights, which are also assumed to be most fundamental, are the *right to exclude* and the *right to use*. The first allows an owner to exclude someone from their property and its usage, while the latter allows the owner to use their property appropriately. One may be inclined to think that one could choose to do with one's property whatever one wants to. But this is, in general, an overstatement. For example, we are typically not allowed to use our property in ways that interferes with fundamental rights of others.

The central problem with hybrid devices is often lurking around ownership rights and it is not always clear how to analyze and solve the conflicts resulting from the tension of those rights and the role embedded software plays. For example, while you in fact buy and, thus, own the hardware part of your hybrid devices, you typically neither buy nor own their software components: You rather license them in terms of some license agreement – even though many don't realize that fact. The distinction between owning and having licensed something is not a purely semantic issue. The rights (legally, but also morally speaking) that come with ownership are typically much richer and more far-reaching than the rights which are granted by license agreements. You are under a restriction imposed by a license agreement – those pages you never read when downloading, buying or installing some software. For instance, you are often prohibited to resell software you bought, because you never really *bought* the software but only a permission to install and use it.[12] But since you are typically allowed to sell what you own, the questions arises: Are you allowed to sell your hybrid devices? You own them, thus you have permission to sell them. But the software within is only licensed by you and the agreement may well be such that you have not the permission to sell the software. But whenever you sell such a device, you factually sell the embedded software as well. So, do you have the permission to sell your property or do you not have that permission? If you don't have this permission, this goes far beyond the limitations we already experience in the field of software reselling: E.g., imagine, you are no longer allowed to sell your car, because nowadays cars do contain many embedded systems, including embedded software. There is an ongoing discussion how to handle cases like this given the laws applicable. But also it is discussed in more general and philosophical terms how reasonable laws *ought* to handle such cases to be in accordance with natural, moral rights – rights often assumed to be more fundamental.[13]

While this is not the conflict-afflicted feature of those hardware-software hybrid entities this paper is concerned with, it points to the fact that there is

[11] The idea is at least as old as the foundation of modern law, e.g., it can be traced back to [2]. It can be found more informatively stated in modern terms in the very influential [7]. For a detailed discussion and summary of those two rights see [5].

[12] See [11].

[13] For a broad overview of different aspects see the Owner's Rights Initiative, an organization that fights for adequate rights: http://www.ownersrightsinitiative.org/.

a pending conflict that can emerge from rights connected to ownership and the role software plays in hybrid devices. In the case of software doping, I argue, this is the case, too.

Even if owners are not allowed to choose to do with their property whatever they desire to do, at least in typical cases there is an owner's right to use her property in *some* way. If in a case where the owner of some property is allowed to use her property in a certain way, someone or something hinders the owner to exercise her right, then this violates her right and, hence, is at least pro tanto morally wrong. I argue that software doping does exactly that: It hinders owners from exercising their rights to use and is, therefore, pro tanto morally wrong.[14] The argument has two steps.

First, notice that if an owner wants to use one of her devices $D$ in a way compatible with $D$'s proper functionality, then this doesn't interfere with rights at least equally as fundamental as the rights of others – at least not in general: If the proper functionality of some devices of a certain type were in general violating fundamental rights of others, this output, by definition of proper functionality, was reasonably to assume or be expectable in a transparent way. This, however, would make it extremely implausible for devices of this type to be allowed by any acceptable legal system, because an essential feature of such morally acceptable legal systems is that it prevents any systematic violation of fundamental rights of persons. Since, I assume such legal systems to be in place in the relevant cases, I am therefore allowed to conclude that, in general, it holds that if an owner wants to use some freely available device $D$ in a way compatible with $D$'s proper functionality, then this doesn't interfere with rights at least as fundamental as their rights of others. This observation is compatible with the existence of cases where one can use $D$ in a way compatible with its proper functionality that nevertheless conflicts with such rights of others, but then – for the above given reasons – this is in exceptional circumstances. For example, you can drive your car on a proper street, but crash into a group of children passing the road. Then you used the car in its proper way, but under special and inauspicious circumstances.

Secondly, realize that by its very definition, software doping modifies someone's device's behavior intentionally in a way that isn't in line with the owner's interests for serving the interests of the manufacturer instead. Therefore, whenever there is software doping, it by definition hinders the owner from exercising the owner's right just established. By neglecting the justified interests of owners, software doping violates rights of owners. Hence, software doping is necessarily (by definition) pro tanto morally wrong.

[14] In terms of standard normative ethical theories, what is wrong is typically an action or a type of action. I therefore use "software doping" as an ellipsis for the act of implementing software doping. However, there is an influential idea in computer ethics (the so-called *embedded value approach*) that holds that computer systems and software are not morally neutral. For a deeper discussion of that topic see [3].

### 3.2 The All Things Considered Wrongness of Software Doping Relative to Normative Theories

Let us accept the conclusion of the last section, that is, accept that software doping is pro tanto morally wrong. But is it reasonable to hold the stronger claim that software doping is generally wrong all things considered? This depends, as I explain, on two aspects: the normative background assumptions and (possibly empirical) facts about cases of software doping. While the latter is a rather complicated issue not easy to generalize and at least in some cases subject of empirical science and not of philosophical investigation, the next subsection nevertheless hints at a certain direction motivated by a recent example, the VW emission scandal. This subsection, however, sheds light on the role of the mentioned normative background assumptions.

In normative ethics we find at least three families of theories: *Consequentialist* theories evaluate the normative status (being demanded, permitted, or forbidden) respectively the moral status (being morally right or morally wrong) solely on the qualities of the (actual or reasonably expectable) outcomes of available options, e.g. actions; *Deontologist* theories judge actions based on their compliance with certain rules. Typically duties, obligations, and rights play central roles in those theories. Finally, there are so called *virtue ethics*, which rather accent the role of the agent's character than the properties of her actions. All three families are 'densely populated' by a variety of theories, all basically claiming to be the correct theory of the morally right and wrong, the permitted, the forbidden, and the demanded.

Since it isn't even clear what kind of agent we are confronted with in the case of software doping, I focus on consequentialist and deontologist theories.[15] They are the most frequently held, anyway.

The idea of pro tanto wrongness in contrast to all-things-considered wrongness is compatible with and commonly accepted by all three families of normative ethical theories.[16] Also the connection between ownership and certain rights can be agreed upon by all three theories, albeit for very different reasons. A consequentialist, e.g., may accept such bundles of rights coming with ownership for pragmatic reasons and judge the violation of such rights as morally bad. It becomes clear that everything I showed to this point is, loosely speaking, independent from the choice of any kind of ethical theory.

[15] Is it the manufacturer, a single software engineer, a group of engineers, or the managers of the company? It is even more hazy whether these agents, whoever they might be, can be addressees of virtues.

[16] Immanuel Kant seems to be a special case: According to his specific deontological ethical theory around the categorical imperative, acting such that you don't meet one of your duties means you have acted wrongly, period. Therefore, even if you lie to the end of saving someone's life, you are acting wrongly by doing so, since according to Kant you are under a moral duty to speak the truth. However, it turns out that nowadays even Kantians, i.e., philosophers that agree with Kants thoughts, don't subscribe to the original – radical and rigorous – view anymore.

That being said, the theories don't always agree in their moral evaluations. They especially disagree when it comes to questions of all things considered wrongness: It can be the case that, while theories of different families of normative ethical theories agree in the pro tanto wrongness of some act, they disagree whether the act is morally wrong all things considered. E.g., if we can save two innocent persons from a painful death by killing a third one – someone that otherwise would have remained unaffected from our decision – this is indisputably pro tanto wrong according to most if not all plausible normative ethical theories: After all, we would kill an innocent person. However, according to many consequentialist theories such an act would be nevertheless overall morally right (if there are no better alternatives, of course), because the outcome of two innocent persons being alive and unharmed is better than the alternative outcome where only the third remains so. According to most deontologist theories, however, it would be wrong to kill one, even in order to save two, period.

Hence, the question we are concerned with comes down to the following: Can we reasonably assume in general that (i) other duties, obligations or rights are in place that 'overwrite' the pro tanto wrongness of software doping or that (ii) there are other good outcomes to be expected in such cases that 'outweigh' this pro tanto wrongness? Whether (i) or (ii) is pertinent depends on the choice of deontologist or consequentialist normative ethical theories respectively as background assumption.

### 3.3 A Clear Example of Wrongful Software Doping: The VW Emission Manipulation

Recently, VW was caught using a software version of a *defeat device*. A defeat device is defined as "an auxiliary emission control device [...] that reduces the effectiveness of emission controls under conditions that the locomotive may reasonably be expected to encounter during normal operation and use."[17] The software is able to detect laboratory emission testing and to adapt the engine such that the classical empirical evaluation for singular points under laboratory conditions become insufficient. The software was intentionally programmed for this purpose and used in Turbocharged Direct Injection (TDI) diesel engines during model years 2009 through 2015, affecting about half a million cars in the US and about eleven millions worldwide. It thereby constitutes a clear example of software doping, because, e.g., it serves the interests of the manufacturers – as long as not detected –, since it means a competitive advantage for them, and frustrates the interests of the users/owners, namely not being lied about the features of their car.

Most fundamentally, the $NO_x$ outputs were adapted to meet US standards during regulatory testing, while in real-world driving emitting up to 40 times more $NO_x$. As a recent study, [1], suggests, approximately 59 estimated premature deaths have been caused by the excess pollution produced between 2008 and

[17] According to Code of Federal Regulations Title 40 - Protection of Environment by the Environmental Protection Agency: https://www.gpo.gov/fdsys/pkg/CFR-2015-title40-vol33/xml/CFR-2015-title40-vol33-sec1033-115.xml.

2015 by vehicles equipped with the defeat device – in the U.S. alone. Current investigations of the German Federal Motor Transport Authority (Kraftfahrt-Bundesamt, or KBA) may reveal further irregularities and similar cases of software doping involving Mercedes, BMW, Renault, and others.[18]

Maybe in the VW case, software doping was at least a rational choice, since it allowed the company to sell more cars. But even that may be disputed for good reasons.[19]

Since software doping in many cases aims at circumventing regulations and laws and because normally, acceptable regulation and laws aim at the public good, one can hold for good reasons that cases of software doping normally (under normal circumstances) don't aim at the public good. Obviously, in general, manufacturers have reasons to do what is *not* in the interest of the society: often their interests are contrary or even contradictory to the public interest. Software doping, thus, often causes harm additionally to the rights violation it necessarily causes. It need not be as harmful as the VW emissions, which in fact killed people, but it will often cause more harm than benefits compared to alternative options. Therefore, in consequentialist terms, we can easily and plausibly prima facie[20] ascribe all-things-considered wrongness to software doping in general.

From a deontologic point of view we must look out for duties of VW that may speak in favor of deception. True, VW was under a duty of success towards its shareholders. True, the company was under a duty of assuring its 'survival' towards its employees. And perhaps the only way to act according to these duties was applying the defeating device software into their diesel engines. But by no plausible view can those duties justify the eleven million times committed right violations of owners and even less they can justify their moral misconduct towards the rights of those seriously harmed or even of those killed by the illegal emissions.

Therefore, we can be sure that all cases of software doping are morally wrong even all things considered in light of deontologic views. And even from a consequentialist point of view we can hold such a default view on software doping for good reasons. Still, such theories may allow more space for exceptions.

[18] The data is not yet published, but first media reports suggest so, see http://www.spiegel.de/auto/aktuell/kba-misst-auch-bei-anderen-autoherstellern-erhoehte-abgaswerte-a-1062251.html.

[19] Either the responsible managers at VW thought they wouldn't get caught or they were only interested in short term benefits. Because it was obviously not in their interest to pay a billion dollar penalty – not to mention the awful stock market effects – and loose reputation in magnitudes hard to express in monetary value. But maybe they thought that, even if they were caught, that wouldn't be that bad – for what reasons ever –, or that all possible bad effects might be weighed out by the expectable (competitive) advantages.

[20] This evaluation can only be prima facie since there may well be exceptional cases, where (expectable) positive outcomes weigh out the general badness of a specific instance of software doping.

## 4 Conclusion

I developed a proposal of a definition of "software doping", then analyzed the moral features of the concept. I noted that software doping is necessarily pro tanto morally wrong because, by definition, it is in conflict with certain rights of owners. This leaves room for the possibility that it might nevertheless be morally right all things considered under specific circumstances, relative to some normative ethical theory as background assumption, first and foremost consequentialist theories. I argued, based on the example of the VW emission scandal, that it is reasonable to assume that such circumstances are not in place in general. It is thus also reasonable to think of software doping as something morally wrong as a default.

From this conclusion it follows that for moral reasons alone we ought to overcome software doping and thus ought to develop promising methods that allow to detect software doping, since this seems to be the most promising way to fight software doping from a pragmatic point of view. If the proposed definition is appropriate in its central aspects, such a development of software doping detection methods must proceed in two steps. First, one needs an explicit specification of a device's proper functionality to gain transparency. This comprises the final hybrid device, but surely also all smaller hybrid components with behavior dependent from embedded software. For this, models of the expected input-output-behavior of devices must be introduced. Second, one needs methods that allow for testing actual device's behavior against such specifications. This must be done on a much broader basis than under laboratory conditions, because more or less all realistic circumstances must be considered. In a nutshell: transparency must be achieved with regard to the de facto influences embedded software has on the behavior of modern hybrid devices.

## References

1. Barrett, S.R.H., et al.: Impact of the Volkswagen emissions control defeat device on US public health. Environ. Res. Lett. **10**, 114005 (2005)
2. Blackstone, W.: Commentaries on the Laws of England. Clarendon Press, Oxford (1776)
3. Brey, P.: Values in technology and disclosive computer ethics. In: Floridi, L. (ed.) The Cambridge Handbook of Information and Computer Ethics, pp. 41–58. Cambridge University Press, Cambridge (2010)
4. Cook, T.: A message to our customers. www.apple.com/customer-letter/. Accessed Mar 2016
5. Douglas, S., McFarlane, B.: Defining property rights. In: Penner, J., Smith, H.E. (eds.) Philosophical Foundations of Property Law, pp. 219–243. Oxford University Press, New York (2013)
6. Grenoble, R.: Political protest or just blowing smoke? anti-environmentalists are now rolling coal'. Huffington Post. www.huffingtonpost.com/2014/07/06/rolling-coal-photos-video_n_5561477.html. Accessed Apr 2016
7. Honoré, A.M.: Ownership. In: Guest, A.G. (ed.) Oxford Essays in Jurisprudence, pp. 107–147. Oxford University Press, New York (1961)

8. Moor, J.H.: What is computer ethics? Metaphilosophy **16**(4), 266–275 (1985)
9. Opsahl, K.: EFF to support apple in encryption battle. Electronic Frontier Foundation. www.eff.org/de/deeplinks/2016/02/eff-support-apple-encryption-battle. Accessed Mar 2016
10. Pym, S., (U.S. Magistrate Judge): Order compelling Apple, Inc. To Assist Agents in Search. www.documentcloud.org/documents/2714001-SB-Shooter-Order-Compelling-Apple-Asst-iPhone.html. Accessed Mar 2016
11. von Lohmann, F.: You bought it, you own it: vernor v. autodesk. Electronic Frontier Foundation. www.eff.org/deeplinks/2010/02/you-bought-software-you-own-it-vernor-v-autodesk. Accessed Mar 2016