DECONSTRUCTING DESIGN DECISIONS: WHY COURTS MUST INTERROGATE MACHINE LEARNING AND OTHER TECHNOLOGIES

BY

ANDREW D. SELBST
ASSISTANT PROFESSOR OF LAW

AND

SURESH VENKATASUBRAMANIAN
PROFESSOR OF COMPUTER SCIENCE AND DATA SCIENCE, CENTER FOR TECH RESPONSIBILITY, BROWN UNIVERSITY

AND

I. Elizabeth Kumar
PH.D CANDIDATE IN COMPUTER SCIENCE, CENTER FOR TECH RESPONSIBILITY, BROWN UNIVERSITY.

DECONSTRUCTING DESIGN DECISIONS: WHY COURTS MUST INTERROGATE MACHINE LEARNING AND OTHER TECHNOLOGIES

Andrew D. Selbst[*]
Suresh Venkatasubramanian[**]
I. Elizabeth Kumar[***]

Technologies do not just come about. They are designed, and those design choices affect everything the technology touches. Yet unless a legal question directly implicates the technological design, courts are not likely to interrogate it. In this Article, we use examples from machine learning to demonstrate that the design choices matter even for cases where the legal questions do not involve technology directly. We start by describing formal "abstraction," a fundamental design technique in computer science that treats systems and subsystems as defined entirely by their inputs, outputs, and the relationship that transforms inputs to outputs. We show how this technique causes the resulting technologies to be effectively making claims about responsibility and knowability that competes with courts' own determinations. We further show that these claims are rendered invisible over time. Thus, we argue that courts must unearth—or deconstruct—the original design choices in order to understand the legal claims in a given case—even those cases that do not on their face appear to be about technological design.

There is, of course, a reasonable concern that courts are not capable or are not the best venue to make judgments about technological design. While we agree that courts are not the optimal front-line regulators of technology, we argue that they cannot avoid these questions as technologies begin show up in every type of case—a phenomenon that will only grow with time. But besides being forced to consider technology, courts are actually capable of doing so

---

[*] Assistant Professor, UCLA School of Law

[**] Professor of Computer Science and Data Science, Center for Tech Responsibility, Brown University

[***] Ph.D Candidate in Computer Science, Center for Tech Responsibility, Brown University.

The authors would like to thank Mario Biagioli, William Boyd, Ryan Calo, Michael Froomkin, Jerry Kang, Mark McKenna, Kirsten Martin, Ted Parson, Xiyin Tang, Karen Yeung, Tal Zarsky, and the participants of the 2021 Privacy Law Scholars' Conference, We, Robot 2021, and the UCLA Juniors' Workshop, for incredibly helpful comments on earlier drafts. Thanks also to Ellie Baines, Cecilia Bobbitt, and Kaysie Gonzalez for excellent research assistance.

when motivated to. We demonstrate that in certain cases that clearly tee up technological design, such as products liability, copyright retransmission, and the functionality doctrines of intellectual property, courts have no problem diving in and questioning the design choices, asking what could have and should have been. Where courts can perform analysis in one arena, they can do so in another. Finally, through extended hypotheticals in the areas of negligence, discrimination, and criminal justice, we demonstrate how courts can effectively deconstruct technological design.

INTRODUCTION

Black boxes are ubiquitous in narratives surrounding accountability for technological harms, especially machine learning and artificial intelligence. The black box is usually the villain of the story, the reason that there cannot be accountability for the harms. The black box is a trade secret.[1]

[1] *E.g.* FRANK PASQUALE, THE BLACK BOX SOCIETY (2015); Danielle Keats Citron & Frank Pasquale, *The Scored Society: Due Process for Automated Predictions*, 89 WASH. L. REV. 1, 33 (2014); Sonia K. Katyal, *The Paradox of Source Code Secrecy*, 104 Cornell L. Rev. 1183, 1186 (2019); Frank Pasquale, *Restoring Transparency to Automated Authority*, 9 J. ON TELECOMM. &

It's inscrutable.[2] It's both.[3] But while most analyses focus on the opacity or mystery of the box, few examine the process of blackboxing.[4] This Article is about the construction of the box—the choices that go into it, who makes these choices, how the choices are rendered invisible, and how this all affects the work of courts. While our argument applies broadly to the use of technology in courts, much of our discussion will focus on machine learning and algorithmic systems, both because it is an urgent topic and because those are the technologies we work with and know the best.

What we know as a technology is the end result of a set of design decisions that a developer of that technology went through to come up with a product. In computer science, the process of creating black boxes is formalized and is known as abstraction. While abstraction is a concept that has a standard English usage, it is a term of art in computer science. Abstraction is the process of separating complex systems into isolated components that can be completely described by their inputs, outputs, and the relationship between them, without needing to know anything about the internal operation. It is a design technique to manage information flow and system architecture, without which it would be almost impossible to build systems of any meaningful complexity. In the less formal sense, we all use abstractions in our daily lives—any concept we hold can be considered an abstraction—but the formal process of abstraction is a foundational

HIGH TECH. L. 235, 236-37 (2011) (recounting the origins of using trade-secret protections for algorithms); Brenda Reddix-Smalls, *Credit Scoring and Trade Secrecy: An Algorithmic Quagmire or How the Lack of Transparency in Complex Financial Models Scuttled the Finance Market*, 12 U.C. DAVIS BUS. L.J. 87, 88-90 (2011).

[2] *E.g.* Ashley Deeks, *The Judicial Demand for Explainable Artificial Intelligence*, 119 COLUM. L. REV. 1829 (2019); Sandra Wachter et. al., *Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR*, 31 HARV. J.L. & TECH. 841, 843 (2018); Cary Coglianese & David Lehr, *Transparency and Algorithmic Governance*, 71 ADMIN. L. REV. 1, 4 (2019).

[3] *E.g.* W. Nicholson Price II, *Black-Box Medicine*, 28 Harv. J.L. & Tech. 419, 434 (2015); Charlotte A. Tschider, *Beyond the "Black Box"*, 98 DENV. L. REV. 683 (2021); Emily Berman, *A Government of Laws and Not of Machines*, 98 B.U. L. REV. 1277, 1315 (2018)

[4] A notable exception is David Lehr and Paul Ohm's breakdown of the creation of the machine learning process that functions as a primer for legal scholars. David Lehr & Paul Ohm, *Playing with the Data: What Legal Scholars Should Learn About Machine Learning*, 51 U.C. DAVIS L. REV. 653, 656 (2017).

methodological component of computer science, often taught on the first day of an introductory class.[5]

Abstraction serves to separate a subsystem from the rest of the system, and a technical object from the rest of the world. As such, the choices that go into an abstraction—the definition of inputs and outputs, the specification of the operation that converts inputs to outputs—affect how the world sees the object in question and how relationships to the object are formed. Abstraction also imbues the technology with a viewpoint about what the boundary of the technology is and who should be responsible for which outcomes. As Science and Technology Studies (STS) scholar Madeleine Akrich has observed:

> [C]hoices made by designers take the form of decisions about what should be delegated to whom or what, [meaning] that technical objects contain and produce a specific geography of responsibilities, or more generally, of causes. To be sure this geography is open to question and may be resisted. Nevertheless, it suggests that new technologies may not only lead to new arrangements of people and things. They may, in addition, generate and "naturalize" new forms and orders of causality and, indeed, new forms of knowledge about the world. . . . [T]echnologies may generate both forms of knowledge and moral judgments.[6]

Abstractions are rules that form new relationships between people, technical systems, and their environments. The black box is considered a black box precisely because it is opaque; its purpose is to hide the details of what's going on inside and to insulate those concerned with the internal parts from any external context. When the developer puts an algorithmic system in a black box, she is defining the object, implicitly telling the user that they need not worry about its contents, and telling the world that anything outside the box is not her problem.[7] This communicative aspect is an unavoidable byproduct of creating the technology.

While in computer science, black boxes are the end result of abstractions, "blackboxing" has another important and related meaning in

[5] *See, e.g. 6.001: Structure and Interpretation of Computer Programs Class 1 Lecture Notes*, MASS. INST. OF TECH. (2004), https://ocw.mit.edu/courses/6-001-structure-and-interpretation-of-computer-programs-spring-2005/resources/lecture1webhand/.

[6] Madeleine Akrich, *The De-Scription of Technical Objects*, *in* SHAPING TECHNOLOGY/BUILDING SOCIETY 205, 208 (Wiebe Bijke & John Law eds., 1992).

[7] *See* Kirsten Martin, *Ethical Implications and Accountability of Algorithms*, at 8-9, J. BUS ETHICS (2018) (describing how algorithm design "prescribes the delegation of responsibilities in decisions").

STS. It refers to the social process by which the choices made in technological design are rendered invisible over time.[8] When a technology is new, we understand it by analogy or by reference to its inner workings. We are curious; we poke and prod as we flesh out the new object in our minds, collectively refining our understanding of the technology until a shared understanding of the concept eventually emerges. Then as we all come to accept a technological phenomenon, it becomes commonplace. We learn to use it, to live in a world in which it exists, and for the most part, to accept it as it is. Once an object has been blackboxed, we forget the debates about what does and does not constitute the object and why. It simply exists.

This brings us to the work of courts. Technological systems can be involved in a wide variety of court cases. A plaintiff may be physically injured by a medical practice that uses AI or discriminated against by an automated hiring or lending system. A criminal defendant can be denied release from jail based on a risk assessment. Add in product liability, intellectual property, telecommunications, privacy, or even labor disputes with app-based employers like Uber,[9] and we see that technology can be present in any type

[8] *See* Matthew Shindell, *Outlining the Black Box: An Introduction to Four Papers*, 45 SCI. TECH. & HUM. VAL. 567, 573 (In STS and related fields the black box metaphor "provides a language to talk about knowledge production and in particular how the practices of science and technology obfuscate assumptions about and interactions with the world, while making facts and things seem natural and self-evident.").

This definition was originally offered by Bruno Latour to describe the difference between science in process, and science that has been concluded and established as fact. *See* BRUNO LATOUR, SCIENCE IN ACTION 4 (1987). Notably, the phrase "black box" has had many other meanings over time in different fields. *Rodolfo John Alaniz, Before the "Black Box": The Inputs and Outputs of Nineteenth-century Deep-Sea Science*, 45 SCI. TECH. & HUM. VAL. 596, 598 (2019) ("The definition of "black box" has varied by discipline, and this contestation over the black box's meaning has only grown since the mid-twentieth century.") This includes references as varied as aircraft flight recorders, models of unobservable scientific phenomena in deep space or deep ocean, stages of brain development. *See id* (deep-sea biology as a case study of black box usage); MAURICE NEVILE, BEYOND THE BLACK BOX: TALK-IN-INTERACTION IN THE AIRLINE COCKPIT (2004) (Glossary of Aviation Terms; defining "black boxes" as flight data and voice recorders); Elizabeth R. Petrick, *Building the Black Box: Cyberneticians and Complex Systems* 45 SCI. TECH. & HUM. VAL. 575 (2019) (detailing the development of the black box terminology in cybernetics). We do not intend to arbitrate any debates about black box definitions; rather we merely draw on Latour's definition because it is relevant to our discussion.

[9] *See* Joel Rosenblat, *Uber's Future May Depend On Convincing the World Drivers Aren't Part of its 'Core Business'*, TIME (Sept. 12, 2019) https://time.com/5675637/uber-business-future/ (claiming that its app

of court case. In each of these cases, a court will be tasked with determining whether the outcomes comport with the law, and if someone is injured, who is responsible.

When a court encounters a technology, it encounters it as a black box. This can be either because the inner workings are kept intentionally secret by the developers,[10] or the technology has been blackboxed by time, or both. Either way, by the time the technology appears in court, the epistemic and normative implications of the design choices have been rendered invisible. In this Article, we argue that courts must come to understand that the technological arrangements themselves represent substantive claims that bear directly on legal proceedings.[11] For a court to fail to unpack those claims is to silently adopt the normative and epistemic positions that the technologies represent, giving them an unwarranted legal imprimatur.[12] Design decisions are made by engineers and product managers for the purpose of efficiency, portability, and profit, while courts are in the business of ensuring that conduct matches those normative concerns that society has decided to enshrine in law. Courts must resurface and deconstruct design decisions in order to properly do their jobs.

There are some cases in which courts already do this—specifically cases where technological design is clearly teed up by the legal question, such as products liability or various intellectual property issues. But there are good

[10] The technology industry is famously secretive by default. *See* Sonia K. Katyal, *The Paradox of Source Code Secrecy*, 104 Cornell L. Rev. 1183, 1193 (2019).

[11] We are, of course, far from the first scholars to argue that technology has normative implications with implications for law. For example, both Joel Reidenberg's "lex informatica" framework and Larry Lessig's *Code* discuss the ways in which technologies can be designed to implement social goals either in concert with law or as workarounds to it. *See* Joel R. Reidenberg, *Lex Informatica: The Formulation of Information Policy Rules through Technology*, 76 TEX. L. REV. 553 (1997-1998); LAWRENCE LESSIG, CODE AND OTHER LAWS OF CYBERSPACE (1999). With respects to courts specifically, Andrea Roth has argued that where machines are used to establish facts, attention must be paid to the fact that machines are designed and can produce errors, and they should be critically analyzed. Andrea Roth, *Machine Testimony*, 126 YALE L.J. 1972, (2017). In her article, Roth explores various avenues within evidence law to safeguard against the problems of machine testimony. *Id.* at 2022–51. Our focus differs from prior approaches in that we primarily take aim at technologies that do not appear on their face to contribute legal effect and we argue for a conceptual shift—that courts must recognize that they do so.

[12] Cf. Danielle Keats Citron, *Technological Due Process*, 85 WASH. U. L. REV. 1249, 1297 (2008) (noting the rule of law problems that arise from delegating policy decisions to code and coders).

reasons to believe that courts will not by default unpack these technological assertions, and in particular that they fail to do so when technological questions are not immediately salient. The first reason is that even in some recent cases that *did* center technology, courts have shown little desire to question it. For example, take *State v. Loomis*,[13] the most famous case on risk assessments algorithms in sentencing. Though the court recognized the centrality of the technology,[14] and though it might have pressed the creator on questions about the training data or other aspects bearing on the tool's accuracy, it evinced no interest in critically evaluating the (proprietary) tool.[15] Instead, while noting that the tool could contain errors, the court held that a sentencing court should use the tool, but its result could not be "determinative."[16] That is, the court accepted the tool as is, and crafted a (notably hollow) legal rule to compensate for its flaws rather than seek to question the design choices of the tool itself. Amazingly, as Rebecca Wexler has noted, the court even reasoned that the proprietary nature of the tool posed no problem *because* the court lacked access to its design in addition to the defendant.[17] In the same article, about an emerging trade secrets evidentiary privilege in criminal cases, Wexler describes a number of other cases where courts accept untested trade secret claims because they did not deem the specifics of technology itself important enough to defendants' cases.[18]

The second reason is simply that when it comes to technology, judges and juries are lay people, and the lesson of the blackboxing concept is that unless people have been trained to always examine technology with a critical eye, they do not. There is no reason this would not be as true of judges and juries as anyone else. Society has blackboxed technologies, and society includes judges and juries.

[13] 881 N.W.2d 749 (Wis. 2016).

[14] *Id.* at 242 ("Use of a particular evidence-based risk assessment tool at sentencing is the heart of the issue we address today.")

[15] *See id.*

[16] *Id.* at 274.

[17] *See* Rebecca Wexler, *Life, Liberty, and Trade Secrets* 70 STAN. L. REV. 1343, 1346 (2018).

[18] *See id.* at 1392–95; People v. Superior Court (*Chubbs*), No. B258569, 2015 WL 139069, at *6, *9 (Cal. Ct. App. Jan. 9, 2015) (holding that source code need not be disclosed to defendant because there has been no showing that it is "relevant and necessary"); State v. Bernini, 218 P.3d 1064, 1067-69 (Ariz. Ct. App. 2009) (finding that defendants failed to establish "substantial need" for breath test machine source code); State v. Bastos, 985 So. 2d 37, 42-43 (Fla. Dist. Ct. App. 2008) (breath test source code not sufficiently "material" to compel production).

The third reason is related to the second—even if they think to do deconstruct the technological claims, judges often believe they lack the tools to do so. Even in the midst of the "techlash," our society credits technology with a certain mystique, acting as if engineers and Silicon Valley insiders alone possess the knowledge that allows them to comment on technology. We see this repeatedly play out in the press, where, for example, although various outsiders in academia and civil society have raised the alarm about social media or AI for years and have repeatedly called for regulation of AI,[19] the concerns only garner major media attention when ex-Googler Tristan Harris makes a documentary about it[20], ex-Googler Geoffrey Hinton, the "godfather of AI", comes out against AI,[21] or OpenAI CEO Sam Altman calls for a licensing organization for large language models and is invited by Senators to administer the rules for this organization.[22] Judges, like all lawyers, are certainly no *less* prone to the idea that they are not equipped to understand technology; rather, lawyers' aversion to anything technical or scientific is well known.[23] In one Seventh Circuit case where a litigants' and the lower courts' attention to scientific detail would have been important, Judge Posner noted lawyers' aversion to anything technical, lamenting in the record that "it's increasingly concerning, because of the extraordinary rate of scientific and other technological advances that figure increasingly in litigation."[24]

[19] [[cite]]

[20] *See, e.g.*, Devika Girish, *'The Social Dilemma' Review: Unplug and Run*, N.Y. TIMES (Sept. 9, 2020) (sounding an alarm about social media in a review of "The Social Dilemma")

[21] *See, e.g.*, Cade Metz, *'The Godfather of A.I.' Leaves Google and Warns of Danger Ahead*, N.Y. TIMES (May 1, 2023).

[22] Senate Judiciary Committee hearing on the Oversight of AI, May 16, 2023. https://www.judiciary.senate.gov/committee-activity/hearings/oversight-of-ai-rules-for-artificial-intelligence. Rush transcript at https://techpolicy.press/transcript-senate-judiciary-subcommittee-hearing-on-oversight-of-ai/.

[23] One recent expression of this aversion came from Chief Justice Roberts in the oral arguments of *Gill v. Whitford*, 138 S. Ct. 1916 (2018). Faced with math that explained how bad the partisan gerrymander at issue was compared to a more neutral district map, the Chief Justice dismissed it, complaining that based on his "educational background" he only saw it as "sociological gobbledygook." Tr. of Oral Argument at 40, *Gill v. Whitford*, 138 S. Ct. 1916 (2018) (No. 16-1161) (Roberts, C.J.)

[24] Jackson v. Pollion, 733 F.3d 786, 788 (7th Cir. 2013); *see also id.* ("The discomfort of the legal profession, including the judiciary, with science and technology is not a new

For these reasons, we believe courts are unlikely to deconstruct technology, even though it is necessary. There is one important limitation to our claim: we do not present a definition of technology here. It is a famously tricky thing to define, and we do not seek to wade into such a debate. Certainly for some definitions of technology, these observations do not hold. For example, both pencils[25] and the corporation can be considered types of technologies.[26] The former is mundane and simple enough to not be scary, and the latter is a legal technology that judges would feel fully comfortable dissecting. Roughly speaking, our focus in this Article is on those technologies that are associated with Big Tech, or the "technology industry," though that is not usefully precise these days.[27] We do believe the claim that courts must unpack technology is true for technology more broadly defined than just ML and AI, but there is certainly some limit. This limitation does not really trouble our argument, however. We argue that courts should treat cases involving the technologies that we *are* concerned with as they do any other case that doesn't have such technologies. So if a given technology is on the side of the line for which our argument doesn't apply, then that is precisely because the court is already likely to treat it as we want them to treat the technologies we are discussing.

Ultimately, we see three main challenges to overcome to get courts to deconstruct technology: lack of awareness that they need to do so because the technical details do not appear salient, a concern that courts lack the institutional competence to do so, and—once courts have accepted deconstruction as necessary to their role—a concern that they would not

phenomenon. Innumerable are the lawyers who explain that they picked law over a technical field because they have a 'math block'—'law students as a group, seem peculiarly averse to math and science.'" (quoting DAVID L. FAIGMAN, ET AL., MODERN SCIENTIFIC EVIDENCE: STANDARDS, STATISTICS, AND RESEARCH METHODS, at v (2008, student ed.))); Peter Lee, *Patent Law and the Two Cultures*, 120 YALE L.J. 2, 4 (2010) ("As a general matter, lawyers and science don't mix."); DAVID L. FAIGMAN, LEGAL ALCHEMY: THE USE AND MISUSE OF SCIENCE IN LAW, at xi (1999) ("[T]he average lawyer is not merely ignorant of science, he or she has an affirmative aversion to it").

[25] *See* Eric H. Voice, *The History of the Manufacture of Pencils*, 27 TRANSACTIONS OF THE NEWCOMEN SOCIETY 131 (1949).

[26] *See* CHRISTOPHER M. BRUNER, THE CORPORATION AS TECHNOLOGY (2022).

[27] In recent years, many business publications have repeated the adage that "every company is a technology company." *See, e.g.*, Stephanie Stone, *Why Every Company Is A Technology Company*, FORBES (Jan 23. 2017), https://www.forbes.com/sites/forbestechcouncil/2017/01/23/why-every-company-is-a-technology-company/

know how to do it. This Article proceeds in three parts, each roughly corresponding to one of these challenges.

Part I is about black boxes and blackboxing. It aims to explain how design decisions end up creating invisible normative contentions that should matter to a court. Part I begins by defining formal abstraction, going through successively complex examples. Along the way, it explains the consequences of abstraction choices, including the creation of a boundary between the object and everything outside, as well as defining lines of responsible for the operation of different parts of a system. The Part discuss the reasons that certain abstraction choices tend to be made and how they differ from legal considerations, before finally explaining that due to blackboxing over time, they eventually become hidden after the technology is completed, sold, and used.

Part II responds to the institutional competence concern by reviewing areas of law where courts already deconstruct technologies, working through examples of products liability cases, copyright retransmission cases, and the functionality doctrines in copyright. What our examination of these cases demonstrates is that when the law clearly calls for it, courts already perform the kind of analysis that we expect, analyzing design decisions rather than taking them for granted. This, in turn, suggests that the reason they do not in cases that do not tee up technology directly is not institutional competence but a lack of apparent salience, or perhaps confidence.

Finally, Part III offers examples of the desired deconstruction in action. It walks through several extended hypotheticals, with fact patterns that include algorithmic systems in the areas of negligence, discrimination, and criminal justice. The hope is that by working through the hypotheticals, the results of questioning design choices can become clear, and a set of questions that courts should seek to ask can begin to emerge. Ultimately, we conclude that for this practice to become commonplace, we must develop a regular set of questions to aid courts in deconstructing design, so that they can engage with the technologies that appear in front of them and reach better decisions overall.

I. ABSTRACTION AND DESIGN DECISIONS

In computer science, the process of system design uses technique called abstraction. Abstraction is a formal process that creates boundaries between the inside and outside of a "black box"—a technical component of a system. Abstraction also has the effect of defining the system as an object that exists separately from the rest of the world, and delineating lines of responsibility: the designer is responsible for—and only for—what happens inside the abstraction boundary, and the user should never have to know how the inside works. In this Part, we describe formal abstraction processes, the ways in which abstraction choices can be read as factual claims about the boundaries of a system and normative claims about where responsibility lies, and how the choices become invisible over time.

A. Formal Abstraction

Abstraction is perhaps the most foundational idea in computer science, and more broadly in the design of digital technology. It is the intellectual foundation that allows a designer to understand a system in terms of its smaller components, building up to incredible complexity, by restricting her view to a particular part of the system any time. Formal abstraction is the technique of specifying a system solely in terms of a set of well-defined inputs, a set of well-defined outputs, and the relationship that transforms inputs to outputs. The beauty and the power of formal abstraction is that one need not know—and indeed should not care—how the transformation from inputs to outputs occurs, as long as it works.

For the simplest of examples, imagine a program `square` that computes the square of an integer. Here, the input is an integer *x*, and the output is a number *y* which corresponds to the input *x* squared. The module could be represented in code in a number of ways. Here are two, where the function `abs` computes absolute value:

```
1) define square(x) {
       y = x * x;
       output y
   }

2) define square(x) {
       y = 0;
       for i = 1 to abs(x):
           y = y + abs(x);
       output y
   }
```

These snippets of code will both compute $y = x^2$ for any integer *x*—the first by multiplying *x* by *x* and the second by adding the absolute value of *x* to itself that many times. The point of abstraction is that someone can use the `square` module or subroutine, know that it will return the value they seek, and never have to know what the code inside looks like, as long as it reliably computes the function it is supposed to on the inputs it is designed for. That is powerful. Nonetheless, the *developer* of `square` might care very much about how it is implemented; one version might be more efficient in terms of compute power or time, or one might be harder to debug.

These abstractions are layered into what we call systems. Notice that the second definition of square employs within it the abstraction `abs` that computes the absolute value of an integer. The creator of `square` does not need to think about how `abs` was made – they merely need to know what the function produces from a given input. Equally important is that the creator of `abs` does not need to know what it will be used for. The context, too, is abstracted away. This is how abstractions become subcomponents in larger systems. It's abstractions all the way down.

When a developer encounters a problem to solve, their first step is therefore to identify the necessary inputs, outputs, and the transformation with the desired properties. This becomes, in computer science parlance, the "problem definition."[28] This defines what the developer is responsible for in that moment. Subsequent activities focus on methods (or "algorithms") to transform the inputs to the desired output as effectively as possible. If at any point in the development process the specification of inputs and outputs change, that becomes a new problem, and the cycle starts again. The decisions about what constitutes an input and what constitutes a desirable output are the "abstraction choices" made as part of the design of the resulting system.

These choices also represent a boundary between the interior of a system and its exterior. For the designer of the system, the processes that lead to the production of the inputs as well as the processes that follow after the production of the outputs are not considered part of the problem space and are disregarded. (The creator of `abs` need not know that someone at a later time will come along and use it to make `square`.) Correspondingly, for a user of the system, the internal design choices used to construct the system are abstracted away: the user does not need to know how the input is transformed to the output as long as it does so correctly. (The creator of `square` need not know the choices that went into making `abs`.)

[28] [[cite]]

The math examples above can illustrate the concept of abstraction, but they too simple to demonstrate the stakes of abstraction choices. For a more tangible example, consider the abstraction of a coffee maker.[29] A reasonable abstraction of the problem (making coffee) is to assume as input water and coffee beans, and as output a cup of coffee. This problem definition does not concern itself with where the water came from, whether the beans were sourced from an organic grower, or even what size the user's mug is; any issues concerning the input or output conditions are not the designer's problem. Conversely, a user of this coffee machine operates only on the outside of this system. For the user to brew a delicious cup of coffee, "you need to provide water and coffee beans, switch the machine on and select the kind of coffee you want to get."[30] The user need not know anything about how the machine works: someone else worried about that and created a coffee machine that now acts as an abstraction and hides all these details.

There are several consequences to this abstraction. As stated above, it allows the designers to think about the complex system in a way that allows it to be built more reliably, or even built at all. In this Article, we focus on two specific conceptual consequences of creating this abstraction. One is an epistemic result. We now have a concept labeled "coffee maker" that we did not have before the abstraction was created. We can now refer to the coffee maker as a technical object and we can understand it as one. The conceptual space that the object takes up is delineated by its abstraction boundary. The internal parts of the coffee machine, such as the heating element or the mechanism to move water from the tank to the coffee grinds, are all internal to the abstraction. The rest of the world is external. The machine is a conceptual boundary. The second effect is a division of responsibility. What is inside the machine is different from what is outside. If coffee comes out badly because the heating element doesn't work well, the manufacturer is to blame; if a user inputs foul-tasting water, then it's not the manufacturer's problem.

The specific abstraction of a problem is a choice that has consequences, which can even be seen in this simple example. Consider the abstraction boundary for the coffee machine, between the user providing the

[29] This example too has been used elsewhere. *See* James W. Malazita & Korryn Resetar, *Infrastructures of abstraction: how computer science education produces anti-political subjects*, 30 DIGITAL CREATIVITY 300 (2019); Thorben Janssen, *OOP Concept for Beginners: What is Abstraction?*, STACKIFY (Nov. 23, 2017), https://stackify.com/oop-concept-abstraction/.

[30] Janssen*, supra* note 29.

supplies and the machine that uses them. It may seem natural or obvious, but it's not. For example, there is a difference between a coffee machine that takes in pre-ground coffee and one built with a grinder that can takes in whole beans. Conceptually, whether the grinding is a function internal or external to the coffee machine changes, whether the grinder is a component of the machine changes, and the responsibility for the quality of the grind changes with it as well. Another example is how strictly the inputs are defined. If a coffee machine abstraction only accepts filtered water as an input, then if tap water is used and the coffee tastes bad as a result, that falls on the user; but if the input is merely defined as "water," then the coffee maker manufacturer can potentially be blamed for the bad tap water coffee. Finally, the output must be specified as well. Some coffee machines dispense coffee into a mug provided by the user, and some provide their own pot that the user can later dispense from. If the coffee maker dispenses into a mug, the space provided will dictate what types of mugs can be used, so the mug shape becomes another input to the abstraction. Basically, any abstraction can be created, and many versions can be defended, but the abstraction choices change what counts as the object and who bears responsibility for different outcomes.

This brings us to machine learning. Portable forms of problem definitions and abstraction choices make up the foundations of machine learning technology. For instance, the "supervised learning task", or the problem of building a predictive model, can be solved in many different ways, but it is defined by a particular abstraction:[31] The input to the task includes a training data set, or a list of instances *X* of some data domain, along with exactly one output label *y* for each instance. It also includes a specification for the set of valid solution functions (for example, "linear functions" or "decision tree"), called a hypothesis class. The output of the learning task is a model that maps *X*'s to *y*'s and is the type of function that the hypothesis class dictates. The function should accurately calculate predictions on unseen data. For example, the domain of *X* could be "emails," and each email could be assigned a label *y* of "spam" or "not spam." This setting, in which outputs must be discrete, is an instance of the supervised learning task called classification. Of course, there is much left unspecified in this abstraction about how to solve the supervised learning task. Which algorithms should we use to search the hypothesis class for a solution? What types of functions should be considered? How should a function's performance be estimated or

[31] GARETH M. JAMES, ET AL., AN INTRODUCTION TO STATISTICAL LEARNING 6 (2013).

evaluated? What assumptions must be made about the probabilistic distribution of the incoming data? These are all parts of the abstraction. In real examples, fully specifying an abstraction can be quite complicated, but abstractions nevertheless provide utility.

Popular solutions to the supervised learning task are implemented by the scikit-learn Python package.[32] Each solution is implemented by different functions within the package but can be presented to a programmer looking to build their own machine learning model through a unified framework. For instance, though a linear solution and a decision tree solution are trained based on differing assumptions about the data, each has a fit method which only takes in training data. Here, we can see how simple it is to train the two models with very similar lines of code.

```
1) lr_model = linear_model.LinearRegression()
   lr_model.fit(X,y)

2) tree_model = tree.DecisionTreeClassifier()
   tree_model.fit(X,y)
```

Because this abstraction is so portable, most solutions to the classification problem can be "plugged in" to many real-life problems without much extra work, such as determining whether an x-ray image of a lung contains a cancerous tumor, or which character in the English alphabet an image of a handwritten character displays. Thus, scikit-learn saves the programmer the work of actually implementing these classification solutions, and in turn guarantees that the results will be correct according to their specifications. The developer is thus alleviated of the responsibility of performing that task. Importantly, the package functions used to train the model remain conceptually separated from the model itself.

While scikit-learn takes much of the responsibility of building a good machine learning model out of the hands of the programmer, the documentation and behavior of the technology makes it clear that there are elements of the resulting technology that are outside of the responsibility of the package. Because the training data is an input, the package assumes that data is correct; the standard classification learning task as understood by

[32] SCIKIT LEARN, http://scikit-learn.org/ (last visited Feb. 11, 2023).

mathematicians does not account for issues of measurement or corruption.[33] It is the programmer's job, not the package's, to validate this. It is also the programmer's job to select which implementation of the classification problem to use; since they may perform differently for different problem settings, it is up to the programmer to pick one that is appropriate. For more advanced users, many knobs are available for a programmer to adjust how the learning procedure is implemented; however, the developers of scikit-learn provide extensive range of default settings, implicitly accepting some responsibility for the model in that scenario.[34] In this case, this responsibility for reasonable default behavior was also noted explicitly by the developers of scikit-learn. Another way of understanding this is that the scikit-learn package abstracts away the *learning task*, but it is only a component of the larger system abstraction, with its own definitions chosen by the developers. Of course, there are other ways to build machine learning systems; scikit-learn is merely one popular option that also offers a relatively clear division of responsibilities.

B. Abstraction Boundaries and Choice

Though abstraction boundaries imply factual and normative claims about what the object is and where lines of responsibility fall, from the engineers' perspectives, they are not making such claims at all. Rather, the choice of abstraction boundary is a design choice governed by concerns such as efficiency and portability, economic and marketing concerns, and legal requirements, where they exist. It is important to recognize as well that the claims in question are implied by the technical arrangement itself, not by the intentions or statements of the actors involved.[35] Nonetheless, the abstraction choices do encode considerations that were taken into account.

[33] HAL DAUMÉ III, A COURSE IN MACHINE LEARNING (2022), https://ciml.info. Though classifiers generally rely on good data, there is a more complex area of "robust" classification that explicitly considers the possibility that a small fraction of the training data is corrupted (by an adversary or through a random process). *See, e.g.*, Dimitris Bertsekas et al., *Robust Classification,* 1 INFORMS JOURNAL ON OPTIMIZATION 2 (2019).

[34], Lars Buitinck et al., *API Design for Machine Learning Software: Experiences from the Scikit-Learn Project* (unpublished paper), https://arxiv.org/abs/1309.0238 ("Whenever an operation requires a user-defined parameter, an appropriate default value is defined by the library. The default value should cause the operation to be performed in a sensible way (giving a baseline solution for the task at hand).")

[35] Martin, *supra* note 7, at 6 ("Scripts are durable, and the technology's script becomes independent of the innovator once in use".)

In computer science, choices about abstraction boundaries are often motivated by the ideals of portability and efficiency.[36] One solution that can be "plugged into" a group of similar problems is preferred to the parallel development of multiple individual solutions which share components or similarities. We can even apply these design principles in the coffee machine example. A machine designer might decide that a coffee machine with its own grinder is more "portable" because of easy access to beans, or that a machine that takes in pre-ground beans is more efficient because the designer no longer has the problem of syncing the grinding and brewing processes.[37]

While computer scientists are motivated by the fundamental imperatives of portability and efficiency, they regularly take other factors into account as well. Designers can—and often do—also choose to define their abstraction boundary in such a way as to account for the context in which the tools they develop will be used. The broader computer science community has been actively researching the complexities of human-computer interaction for decades.[38] By the 1980s, the software industry was no longer simply composed of technically trained programmers; individuals were being hired as "usability professionals" to make sure the exteriors of the tools they developed were intuitive and effective for users.[39] Methods to evaluate the usability of a tool range from informal or heuristic evaluations to formally specified inspection methods to empirical methods such as user testing.[40] For this reason, when new technology is introduced, researchers often try to anticipate interactions that could cause adverse outcomes.

Many successful systems built upon machine learning algorithms are, in fact, designed explicitly to account for human interaction; concerns about the user experience are incorporated directly into the interior of the abstraction boundary. Some good examples can be drawn from the field of

[36] *See* Andrew D. Selbst et al., *Fairness and Abstraction in Sociotechnical Systems,* PROC. 2019 ACM CONFERENCE ON FAIRNESS, ABSTRACTION AND TRANSPARENCY 59.

[37] Note here that this determination of efficiency comes from the perspective of the machine maker rather than the world as a whole. Someone still has to grind the beans, but putting the grinding outside the abstraction makes it more efficient for *the maker* because it is someone else's problem. Some efficiency gains may reduce cost overall, while some reduce cost only for the creator of the technology.

[38] *See generally* Brad A. Myers, *A Brief History of Human-Computer Interaction Technology*, 5 ACM INTERACTIONS 44 (1998).

[39] Deborah J. Mayhew, *User Experience Design: The Evolution of a Multi-Disciplinary Approach*, 3 J. OF USABILITY STUD. 99 (2008).

[40] Jakob Nielsen, *Usability Inspection Methods*, PROC. CHI '94: CONF. COMPANION ON HUM. FACTORS IN COMPUTING SYS. 413 (1994).

natural language processing (NLP). Researchers at Amazon developed Alexa Conversations as a framework "for building goal-oriented dialogue systems that is scalable, extensible, as well as data efficient."[41] The paper describing the tool describes how end-users using the eventual dialogue system may interact with the technology; the researchers specifically design to accommodate "natural conversational phenomena like entity sharing across turns or users changing their mind during conversation."[42] They give an example of ordering pizza, where the system can anticipate user correction ("actually, make it a small"). Researchers at Spotify have also studied how to identify user motivations for making nonspecific queries in an interactive dialogue system, in an attempt to better anticipate their actions and needs.[43] Systems are not always designed to work in a vacuum; they can be designed to work well in context.

Designers can also consider the fact that users can learn to use and reason about tools over time. A good example is SepsisWatch, a tool developed for physicians to rank the probability of sepsis, a leading cause of death in hospitals, among patients. Mark Sendak and colleagues studied the integration of the tool into the Duke University Hospital, finding that:

> [N]urses developed expertise and practices over time that contextualized the information displayed in Sepsis Watch, and facilitated the integration of the tool into existing clinical practice. . . . For instance, RRT nurses developed a practice of working outside of the app and opening a patient's EHR chart before calling the ED physician. By reading through a patient's chart, the RRT nurse was preparing to present the full clinical picture and do 'due diligence,' in the words of one interviewee, in anticipation of questions received from physicians.[44]

[41] A. Acharya, *Alexa Conversations: An Extensible Data-driven Approach for Building Task-oriented Dialogue Systems*, *in* PROCEEDINGS OF NAACL-HLT (2021).

[42] *Id.* at 1.

[43] Jennifer Thom, Angela Nazarian, Ruth Brillman, Henriette Cramer, & Sarah Mennicken, *"Play Music": User Motivations and Expectations for Non-specific Voice Queries*, *in* PROC. ISMIR (2020).

[44] Mark Sendak et al., *The Human Body Is a Black Box: Supporting Clinical Decision-Making with Deep Learning*, PROC. 2020 CONF. ON FAIRNESS, ACCOUNTABILITY, AND TRANSPARENCY, 99, 106.

As the developers of SepsisWatch proceed with maintaining the tool, they can now incorporate this additional information about how it is used, potentially resulting in new specifications for the system.[45]

In each of the above examples, the choices of what to incorporate into the design, and how to anticipate user behavior changes the shape of the technology and what the technical arrangement communicates to the user. Even in the context of machine learning systems, abstraction boundaries can be drawn in a number of different ways. Developers can incorporate different amounts of context surrounding a technical task into the design of an algorithm, resulting in an abstraction boundary that takes responsibility for a large and complex problem specification, or they can choose to only address a simpler, purely technical task with a more limited problem specification.

If design decisions are being made that keep certain aspects of the technology hidden from the users, enable the users to use the technology in certain ways but not others, and communicate to the users certain information but not other information, this all has implications for outcomes in court. If someone is hurt by a user of technology, a court must decide who,

[45] The importance of anticipating user behavior relates to the idea of affordances, which law and technology scholars such as Ryan Calo have been writing about in recent years. *E.g.* Ryan Calo, *Modeling Through*, 71 Duke L.J. 1391 (2021); Ryan Calo, *Privacy, Vulnerability, and Affordance*, 66 DePaul L. Rev. 591, 601 (2017). Originating with James Gibson's work in the ecological theory of perception, *see generally* James J. Gibson, *The Theory of Affordances*, *in* PERCEIVING, ACTING, AND KNOWING: TOWARD AN ECOLOGICAL PSYCHOLOGY 67 (Robert Shaw & John Bransford eds., 1977) and popularized by influential designer Donald Norman, the idea of an affordance is a relationship between an object and the actions it enables. DONALD A. NORMAN, THE PSYCHOLOGY OF EVERYDAY THINGS 8 (1988). In more recent work, Norman differentiates explicitly between affordances—that inherently flexible set of all possible uses—and *perceived* affordances—those apparent uses communicated by the design. Don Norman, *Affordances and Design*, jnd.org (Dec. 3, 2018), https://jnd.org/affordances_and_design/. Under this theory, part of the role of design is about communicating proper use; a handle on a door means pull, and a flat metal piece means push. Overall, the affordance concept conveys that the form of a technology a) enables a set of interactions with the technology that were anticipated in design, b) communicates certain expected uses, and c) enables certain unanticipated uses. A corollary is that a user's expectations around what the technology can and will do in response to their actions is also affected by design decisions and the implications of the design itself. (Think back to the coffeemaker. If there's a grinder internal to the machine, the expectation is that it can adequately grind coffee, and the user will load beans instead of ground coffee). Thus, while we do not rely on the concept much here, affordances and perceived affordances could also be seen as relevant to courts' determinations.

if anyone, should bear the liability for the injury. Courts will also have to decide whether an injury is foreseeable and to whom. This will depend on what the design of the technology looks like in the end: what it enabled a user to do, and what it communicated about the product. Design choices imbue the technology with a viewpoint about what the boundary of the technology is and who should be responsible for which outcomes. A user cannot be held responsible for something they could not have changed or any known about, and a designer cannot be held responsible for an injury caused by unforeseeable use cases.

Recall, though, that a technology designer's choices are not usually made with liability in mind. They are made for profit, for efficiency, for portability, or for usability. Just because a designer made certain choices does not mean that those choices are aligned with legal goals that a court aims to support. Therefore, courts must not only understand the design decisions, but must critically evaluate the design of all technology that is involved in a case. The design of the technology essentially makes claims about foreseeability and divisions of responsibility; but those are legal questions that a court must ultimately decide in accordance with legal principles, not design principles, and it can only do that by understanding the choices that were made.

C. The Invisibility of Abstraction Choices

It is not accidental that the black box is a visual metaphor. According to STS scholar Bruno Latour, blackboxing is "the way scientific and technical work is made invisible by its own success. When a machine runs efficiently, when a matter of fact is settled, one need focus only on its inputs and outputs and not on its internal complexity. Thus, paradoxically, the more science and technology succeed, the more opaque and obscure they become."[46] Abstraction choices, once made, often fade into the background and become invisible, rendering an explicit discussion about the design choices difficult.

While abstraction choices are fundamental to the design process, they are often not documented or made explicit in a formal way. Moreover, it is not obvious from inspection of an already-designed system where the abstraction choices were made and how. Yet the choices have a profound impact, by virtue of placing the boundaries, on how the system might be

[46] BRUNO LATOUR, PANDORA'S HOPE: ESSAYS ON THE REALITY OF SCIENCE STUDIES (1999).

expected to behave under "normal" versus "abnormal" conditions, and what "normal" even means.

Consider the current concerns around the use of facial recognition and analysis in various settings like video interviews, remote proctoring, and surveillance. In all these settings, the standard abstraction choices are expressed in terms of a specifical *technical* task: whether a face is being matched to a given target image, a database, or even whether a face is expressing some kind of emotional affect from which intent can be inferred. Importantly, both the source of the facial scans and the purpose of the match are outside the abstraction boundary and therefore ignored. The concept is the technology. But this division is not preordained; it is an explicit choice made in design or marketing. Consider three different applications where one might choose to use a system that matches faces to a database. One system might be in use at an amusement park to allow visitors access to rides. Another system might be in place at a bank to authenticate account holders prior to transactions. A third system might be in used by the TSA to match returning travelers to their entries in an immigration database.

Imagine now that the abstraction boundary had been drawn more broadly to include these specific applications along with the rest of the technology. We would end up with three distinct problems—amusement park access, bank authentication, and TSA-verification. It is quite likely that the operating conditions for the amusement park access system might involve testing for occluded faces, faces wearing painted-on decorations or silly hats, or even sunglasses. Bank authentication systems might expect an individual to place their face in a very structured setting and have a high bar for a match, because there is a fallback option of entering an ATM PIN or some other password. TSA verification systems might be implemented inside a specific device that can be manipulated so that the image of an individual appears exactly within a prescribed frame and can be cross checked against traveler information provided via a boarding pass scan. The systems might have similar underlying technology, or they might not. Today, because we use the concept "facial recognition" constantly, we tend to think of them as applications of the same underlying technology, but if the abstraction boundary had been drawn to include the applications, we would conceptualize them as different technologies completely (albeit with some clear similarities). The difference in thinking of each of these systems as entirely different is that the underlying class of systems known as "facial recognition" never enters our consciousness as a standalone concept. Because the abstraction boundary was drawn as it was, we use that concept instead.

Because the concept of facial recognition is so ingrained, it might be difficult to accept that we would see amusement park access, bank authentication, and TSA-verification as completely different technologies. So let's look at an example that moves in the other direction. Compare Microsoft Word (a word processor) and Visual Studio Code (an integrated development environment where people write code). Both programs that allow a person to input, format, and print out text, have spell checkers, and have autocomplete among many other similarities. Nonetheless, our understanding of each technology is anchored by their application, not the shared underlying features: one is a word processor and the other an IDE. We probably could have written this law review article in Visual Studio Code—and two of us might even have preferred it—but that does not make the IDE a word processor. Despite similar capabilities, they are seen as different technologies.

In both examples, the choice of abstraction boundary fades into the background as we come to accept the technology presented as a conceptual object.[47] This acceptance of the abstraction as the final object and the rendering of the abstraction choices as invisible *is* the process of "blackboxing" as it is understood by STS scholars.[48]

Coming back to the law, in cases that are not facially *about* the technological details, such as negligence, discrimination, and criminal cases, courts will not typically spend time and resources unpacking design choices and understanding how the technology could have been different. This is in no small part because the choices themselves have been rendered invisible by the process of blackboxing. But it is possible to surface them again through a court's normal evidentiary processes. Litigants can ask the designers what they were thinking, and they can access documentation that paints a picture. The challenge is that along with invisibility comes a lack of apparent

[47] *Cf.* GEOFFREY C. BOWKER & SUSAN LEIGH STAR, SORTING THINGS OUT: CLASSIFICATION AND ITS CONSEQUENCES __ (2000) (noting that classifications in general are necessary then recede into the background).

[48] *See* LATOUR, *supra* note 46; Shindell, *supra* note 8.

salience.[49] Once courts realize the details matter, they can begin to ask how to understand them.

II. WHERE COURTS ALREADY DECONSTRUCT DESIGN

Judges are often generalists in law but are not trained in technology. Yet in this Article we are asking them to examine hypothetical alternatives to the technology in front of them, to examine technological design choices and critique them. It is natural to ask at this point whether courts are institutionally competent to perform such an analysis. We offer two responses to this concern. The first is simple: Courts do not have a choice. Cases involving technology are ubiquitous. While cases *about* technology are obvious examples, we are more interested in those cases that involve technology without appearing to be centrally concerned with that technology. They include legal issues as varied as negligence,[50] employment discrimination,[51] criminal law,[52] evidence,[53] administrative law,[54] labor,[55] and surely others. As the last Section demonstrates, if courts don't at least try to analyze the technology at that point, they are not fully understanding the issues in front of them and are thus abdicating their responsibilities. While it is surely preferable that legislatures and administrative agencies serve as the primary

[49] Jack M. Balkin, *The Path of Robotics Law*, 6 CAL. L. REV. CIR. 45, 46 (2015) ("When we consider how a new technology affects law, our focus should not be on what is essential about the technology but on what features of social life the technology makes newly salient."); Rebecca Crootof & BJ Ard, *Structuring Techlaw*, 34 HARV. J.L. & TECH. 347, 391-93 (2021) (discussing legal salience of technology).

[50] Aarian Marshall, *Why Wasn't Uber Charged in a Fatal Self-Driving Car Crash?,* WIRED, (Sept. 17, 2020, 2:55 PM). https://www.wired.com/story/why-not-uber-charged-fatal-self-driving-car-crash/.

[51] Solon Barocas & Andrew D. Selbst, *Big Data's Disparate Impact*, 104 CALIF. L. REV. 671, 693 (2016); Pauline T. Kim, *Data-Driven Discrimination at Work*, 58 WM. & MARY L. REV. 857 (2017).

[52] State v. Loomis, 881 N.W.2d 749 (Wis. 2016).

[53] Lauren Kirchner, *Powerful DNA Software Used in Hundreds of Criminal Cases Faces New Scrutiny,* THE MARKUP (March 9, 2021, 9:59 AM), https://themarkup.org/news/2021/03/09/powerful-dna-software-used-in-hundreds-of-criminal-cases-faces-new-scrutiny.

[54] Elder v. Gillespie, 54 F.4th 1055 (8th Cir. 2022).

[55] United States v. Uber Technologies, Inc. 3:21-cv-08735 (N.D. Cal. 2021)

regulators of AI and other technologies, courts *will* face these cases, and to do their job correctly, they must consider technological design.

Our second response is the subject of this Part. Courts *are* capable of analyzing technical design. We know this because they already do so in cases where the issue of law clearly presents questions related to technological design. Even if they're not great at it, courts attempt the analysis in many of these cases. Often, because judges are not trained in technical disciplines, this discussion comes in the form of a debate about the appropriate analogy or metaphor for a new technology,[56] and while imperfect, such reasoning is just an expression of exactly what we are asking courts to do.[57]

Examples of deconstructing design are most obviously present in patent law, but that is not the most appropriate example because it is the one area where the concern over courts' lack of technical expertise has led to specialization: the Federal Circuit hears nearly all national patent appeals[58] and from 2011-2021, a pilot project was conducted to allow district court judges to specialize in patent cases, though with mixed success.[59] Outside of patent law, however, technology cases most often show up in courts of general jurisdiction.[60] Thus, in this part we look to a few examples from such courts:

[56] Crootof & Ard, *supra* note 49, at 387 ("All responses to tech-fostered legal uncertainties depend on the use of techlaw analogies."); A. Michael Froomkin, *The Metaphor is the Key: Cryptography, The Clipper Chip, and the Constitution*, 143 U. PA. L. REV. 709, 860–61 (1995) ("The law's first reaction to a new technology is to reach for analogies . . . .")

[57] Crootof & Ard, *supra* note 49, at 388 (arguing that legal actors must both "identify particular characteristics of a technology" and "determine which of those characteristics are relevant or 'salient'" when applying analogies for law and technology).

[58] Paul R. Gugliuzza, *IP Injury and the Institutions of Patent Law*, 98 IOWA L. REV. 747, 760–61 (2012) ("Committee reports on the [Federal Courts Improvement Act of 1982] suggest that the Federal Circuit was needed because patent cases were consuming too much of the regional circuits' time, inexperienced generalist judges were rendering poor quality decisions, and patent law varied greatly among the circuits.")

[59] Ryan Davis, *As Patent Pilot Program Ends, Judges Give Mixed Reviews*, LAW 360 (June 24, 2021, 8:25 PM), https://www.law360.com/articles/1397536/as-patent-pilot-program-ends-judges-give-mixed-reviews.

[60] One might ask why we couldn't just create even more specialized courts that can become more expert on technology. Our answer is that—aside from the difficulties in defining a set of technologies requiring specialized knowledge—technology courts would swallow the judicial system. Technology is ubiquitous and part of what we are pointing out with this article is that technology can show up in nearly any type of case, so all courts need to be ready to handle it. All courts are essentially technology courts.

product liability, copyright retransmission cases, and functionality doctrines within copyright. These examples illustrate how courts can tackle issues of technological design. While our treatment is demonstrative rather than comprehensive, the examples should suffice to demonstrate courts' capacity to do the work.

Leaving aside the question how successful these endeavors are, the recognition that courts already interrogate technological design in certain contexts suggests that the reason they fail to do so in others is not a matter of institutional competence but of salience.[61] Where the technological details appear salient to the outcome, a court will engage in a review of them, as it should. Thus, we are ultimately arguing that courts should expand the set of cases in which they see technical details and design choices as salient questions, to include any case involving technology.

A. Products Liability

A products liability claim applies strict liability for harm caused by defective products,[62] and the central question is whether a product that caused an injury is defective in the relevant sense. There are three types of defects in the products liability tort: manufacturing, design, and warning. Design defect cases are the clearest example of the kind of analysis that we are looking for. A design defect claim argues that the product, as designed, was not reasonably safe, and that the defect in the product caused the injury in question.

The dominant test for design defect is the risk-utility test,[63] which asks whether the risks of a product outweigh its benefits. Courts weigh a number of factors in this analysis, such as the utility of the product, the likely danger it poses, and the obviousness of that danger.[64] The most important factor is the availability of "reasonable alternative designs." Some courts and the

[61] Balkin, *supra* note 49, at 46; Crootof & Ard, *supra* note 49, at 391.

[62] RESTATEMENT (THIRD) OF TORTS: PROD. LIAB. § 1 (1998).

[63] *See* Branham v. Ford Motor Co., 701 S.E.2d 5, 15 & n.11 (S.C. 2010) ("By our count 35 of the 46 states that recognize strict products liability utilize some form of risk-utility analysis in their approach to determine whether a product is defectively designed.").

[64] *E.g.* Armentrout v. FMC Corp., 842 P.2d 175, 184 (Colo. 1992); Denny v. Ford Motor Co., 662 N.E.2d 730, 735 (N.Y. 1995). These factors are often known as the "Wade factors," as they were originally derived from an influential article by John Wade. *See generally* John W. Wade, *On the Nature of Strict Tort Liability for Products*, 44 MISS. L.J. 825 (1973).

Third Restatement of Torts hold that a product is defective only when a reasonable alternative design exists and its omission is what makes the product unreasonably unsafe.[65] This rule has been controversial, with some courts rejecting it as a threshold requirement in the risk-utility test.[66] But even those courts still treat the question as one important factor for the determination of a design defect.[67]

A representative case can help demonstrate how technical design analysis is innately woven into these cases. In *Rider v. Township of Freehold,*[68] the plaintiff, Rider, died when his car collided with a utility pole, and the utility pole "intruded into the passenger compartment."[69] Rider sued the car manufacturer, BMW, among other defendants, on the theory that the car was defectively designed and was thus insufficiently protective.[70] He obtained a favorable jury verdict of just under $7 million.[71] BMW's post-trial motions were denied, including a motion for judgment notwithstanding the verdict, leading to the appeal.[72]

The case is a straightforward example of interrogating design choices. A plaintiff's expert provided evidence that there were "technically feasible, practical and safer alternative" designs available, and a jury found that the lack of them was a substantial factor in the plaintiff's death.[73] In the appellate decision, the court discusses several concrete aspects in the car's design that could have been improved. A sampling: 1) "The front and rear subframes . . . were not connected to one another, creat[ing] a structural gap that allowed penetration"; 2) "[T]he floor pan was made of a thin piece of

[65] RESTATEMENT (THIRD) OF TORTS: PROD. LIAB. § 1

[66] Richard L. Cupp, Jr., *Defining the Boundaries of "Alternative Design" Under the Restatement (Third) of Torts: The Nature and Role of Substitute Products in Design Defect Analysis*, 63 TENN. L. REV. 329, 333 & n.19 (1996) (describing the scholarly controversy over the issue and collecting cites).

[67] *See, e.g.*, Jones v. NordicTrack, Inc., S.E.2d 101, 103 (2001) ("The 'heart' of a design defect case is the reasonableness of selecting from among alternative product designs and adopting the safest feasible one.").

[68] *Rider v. Township of Freehold*, No. A-2319-06T1, 2008 WL 2699805 (N.J. Super. Ct. App. Div. July 14, 2008). *Rider* is not a special case. We chose it as an example because New Jersey is a jurisdiction that roughly follows the Third Restatement, meaning that a showing of a reasonable alternative design is a required part of the plaintiff's design defect case.

[69] *Id.* at *1.

[70] *Id.*

[71] *Id.* at *3.

[72] *Id.* at *1.

[73] *Id.* at *3.

sheet metal that offered no real 'structural resistance' "; 3) "The rocker sections of the car, which run along the perimeter of the underside of the car, were inadequate to prevent intrusion on side impact or compensate for the structural gap left by the separate subframes."[74] The plaintiff's experts opined that the overall structure failed to create a "protective cage," then the experts provided alternative designs that would have formed such a cage and significantly reduced the intrusion into the car.[75] The opinion then went on to discuss other alternative design possibilities.

Reasoning about design choices in the context of ML harms is essentially the same task as examining reasonable alternative designs in products liability. The reason that the court here discusses hypothetical alternative designs is that the law in this area on its face makes the discussion of design choices salient. Because the question of design choices is central to the liability determination in design defect cases, courts cannot avoid the analysis. The court, of course, cannot just come up with the alternatives on its own. In these cases, the court considers evidence provided by experts that the technology could have been different, as well as the tradeoffs associated with designing the technology differently. As we see in *Rider*, relying on expert evidence for this sort of question is something courts are perfectly comfortable with.

Related to reasonable alternative designs is the issue of category liability. Category liability occurs when in order for a plaintiff to establish their design defect claim, they would have to prove that an entire category of products is defective.[76] The Restatement offers a hypothetical example where a plaintiff is injured in a car accident and alleges that his convertible is defective because it is not as safe as a full-size car; it is defective because it is a convertible.[77] Category liability has always been controversial,[78] and by adopting the reasonable alternative design requirement, the Third

[74] *Id.* at *3–*4.

[75] *Id.* at *4.

[76] James A. Henderson & Aaron D. Twerski, *Closing the American Products Liability Frontier: The Rejection of Liability Without Defect*, 66 N.Y.U. L. REV. 1263, 1298–1300 (1991).

[77] RESTATEMENT (THIRD) OF TORTS: PROD. LIAB. § 2 ILL. 9 (1998).

[78] Harvey M. Grossman, *Categorical Liability: Why the Gates Should Be Kept Closed*, 36 S. TEX. L. REV. 385, 386 (1995) (noting "widespread hostility on the part of courts and commentators").

Restatement expressly rejected it, as category liability inherently means that there can be no reasonable alternative design.[79] Again, not all courts agree.[80]

Whether courts permit category liability or not is less important for our purposes than what the debate implies about the relationship between abstraction definitions and liability determinations. Where reasonable alternative designs directly relate to the normative contentions of design choices, category liability relates to epistemic contentions—what counts as the object. Categories of products are not self-defining. As we discuss in Part I, part of the role of abstraction is to define the outer boundaries of the object itself, and to name a concept that corresponds to it. Thus in our common lexicon, "facial recognition devices" is a category of technology, but "amusement park access devices," "bank authenticators," and "TSA-verifiers" are not. The discussion of categories is the same move, though perhaps inverted. While "automobile" might be an abstraction that a plaintiff advances in the Restatement's example to conceptualize the device as defective because it is less safe, a court following the Restatement would instead only be willing to compare convertibles to other convertibles, not automobiles in general, meaning that the court would functionally be choosing the "convertible" abstraction as the correct one, rather than "automobile" in general.

Once again, courts do this in products liability because these issues are salient in those cases. Alex Lemann has a useful discussion of category liability that demonstrates this:

> [N]ot every product is in a "category" of its own. A convertible feels like a category, while a car that lacks a shoulder belt in its rear middle seat does not. Alcoholic beverages, handguns, cigarettes, and above-ground pools have all

[79] James Henderson & Aaron Twerski were the Reporters for the Third Restatement. They had argued earlier for the creation of the reasonable alternative design requirement specifically as a solution to the problem of category liability. *See* James A. Henderson, Jr. & Aaron D. Twerski, *A Proposed Revision of Section 402A of the Restatement (Second) of Torts*, 77 CORNELL L. REV. 1512, 1514 & 1520–21 (1992). It was later incorporated into the Third Restatement.

[80] *See* David G. Owen, *Design Defect Ghosts*, 74 BROOK. L. REV. 927, 933 (2009) ("While some jurisdictions have expressly relied upon the Third Restatement's RAD approach for defining design defects, others have rejected the section 2(b) formulation as imposing unnecessary obstacles to recovery for injured plaintiffs."); Frank J. Vandall & Joshua F. Vandall, *A Call for an Accurate Restatement (Third) of Torts: Design Defect*, 33 U. MEM. L. REV. 909, 926–936 (2003) (discussing states that have rejected the Third Restatement interpretation of design defect).

> been called categories. Just where the line is drawn is not entirely clear . . . In general, plaintiffs are more successful alleging a defect and a reasonable alternative design (and therefore avoiding the problem of category liability) when the challenged aspect of the design is not visible or salient and when its elimination does not significantly change the product's cost or functionality. So, for example, the placement of a gas tank, the use of a particular combustible gas to inflate an airbag, and the design of an ignition switch could all be defects without the doctrines of category liability or reasonable alternative design presenting any obstacle; nobody argues that an airbag inflated by ammonium nitrate is a distinct category of product.[81]

Throughout an analysis of design defects, by the nature of the question, courts engage with technological abstractions. Importantly, though the court has reason to question the technology in front of it, to merely ask about the reasonableness of hypothetical alternatives does not dictate the outcome. Juries will disagree on whether certain alternative designs are reasonable and courts will disagree on what counts as a category of object. These are questions that remain in the hands of legal actors; they are however necessarily informed by attention to the abstractions present in the technology.

B. Copyright Retransmission Cases

Courts have also engaged in this type of analysis in different lines of copyright cases, with mixed success. The cases leading up to *American Broadcasting Companies v. Aereo*[82] are good examples—though the Supreme Court case itself took a different tack. The cases dealt with the right of public performance under the Copyright Act.[83] Aereo was a company that built out a service to retransmit broadcast television signals to subscribers over the internet using small antennas. Based on its understanding of a Second Circuit decision called *Cablevision*,[84] Aereo created a new type of technology to transmit those signals. The resulting cases essentially sought to determine whether Aereo's technological innovation should be seen as complying with

[81] Alexander B. Lemann, *Autonomous Vehicles, Technological Progress, and the Scope Problem in Products Liability*, 12 J. TORT L. 157, 187–88 (2019) (footnotes omitted).

[82] Am. Broad. Cos., Inc. v. Aereo, 573 U.S. 431 (2014).

[83] 17 U.S.C. § 106 (4).

[84] Cartoon Network LP, LLLP v. CSC Holdings, Inc., 536 F.3d 121 (2d Cir. 2008) [hereinafter *Cablevision*].

the transmit clause, or as Justice Roberts suggested at oral argument, merely circumventing it.[85]

To understand the controversy and the role of technological analysis, we need to begin with the Second Circuit's *Cablevision* decision.[86] In the ancient times of 2006, before all media was streaming, consumers regularly used Digital Video Recorders (DVRs) to record programs that were transmitted to them lawfully, with the intent to watch them later. DVRs are the technological successor to video cassette recorders (VCRs), and the private use of VCRs was famously held to be a non-infringing in *Sony Corporation of America v. Universal City Studios, Inc*.[87] Thus, DVRs were not a copyright problem.

In 2006, Cablevision introduced a new service, called Remote Service DVR.[88] This service differed from typical DVRs in that instead of a consumer recording and storing a transmission on a local hard drive, the consumer would direct Cablevision to capture the signal on a server at their facility, and then to transmit it to them once the user has decided to watch it.[89] While under *Sony*, a consumer may always individually record a transmission that was lawfully sent to them, this setup created a second transmission that was a separate "performance" of the work within the meaning of the Copyright Act.[90] The plaintiffs argued that it constituted an a illegal public performance on the part of Cablevision, while Cablevision, in turn, argued both that the performance was non-public and that the copy was actually created by the individual consumer, not by Cablevision.[91] This mattered because if the consumer was the creator of the copy, then Cablevision could not be liable for direct infringement, but only secondary liability for enabling the copy—and the plaintiffs had already disavowed the theory of secondary liability, so Cablevision would be off the hook.[92] The court ultimately held for the defendants: Because each stream was only sent to the one consumer who requested it, it was a private performance, and because the volitional act to

[85] Transcript of Oral Argument at 41, Am. Broad. Companies, Inc. v. Aereo, , 573 U.S. 431 (2014) (No. 13-461)("[Y]our technological model is based solely on circumventing legal prohibitions that you don't want to comply with.")

[86] *Cablevision*, 536 F.3d at 124.

[87] Sony Corp. of Am. v. Universal City Studios, Inc., 464 U.S. 417 (1984).

[88] *Cablevision*, 536 F.3d at 124.

[89] *Id.* at 124–25.

[90] *Id.* at 126.

[91] *Id.* at 125.

[92] *Id.* at 130.

create the transmission came from the consumer, the consumer caused the transmission to happen.[93]

Several years later, relying on *Cablevision*,[94] companies such as Aereo, BarryDriller, and FilmOn X came out with new technology to retransmit broadcast TV signals over the internet, leading to the set of cases that culminated in *Aereo*.[95] The plaintiffs in the cases were the major TV networks—Fox, NBC, CBS, and ABC—and the defendants were the companies like Aereo that created these new technologies. The design of these technologies was, without context, a little bizarre. The technology comprised arrays of miniature antennas, "each no larger than the size of a dime and spaced inches apart."[96] When a consumer elected to watch a broadcast TV program using these services, a single antenna was assigned to them, which then was tuned to the relevant broadcast TV signal, allowing the company to retransmit that signal to the consumer requesting it. This meant that many antennas would be, at any given time, tuned to the same signal and sending it to different customers separately. This was a direct consequence of the design's reliance on *Cablevision*. The companies believed that because the consumer was requesting the retransmission, and it was sent solely to that customer, each transmission was a private performance created by the consumers themselves for the very same reasons that *Cablevision* upheld the provision of Remote Service DVR services.[97]

[93] *Id.* at 139. It's worth noting that a key analogy in this case can also be described as the court choosing between different abstraction boundaries. In deciding whose volitional act makes the copies, between Cablevision and the end customer, the court notes that from the perspective of the person pressing the button, this system looks the same as a VCR: push a button and a copy gets made. *Id.* at 131–32. This description places the entire copying system inside a black box with the content stream and a signal from a user as inputs, and a recorded copy as an output. By choosing to place the abstraction boundary here, the court is declaring that the internal details of how the copy is made are irrelevant to the legal question at hand. Thus, the legal question and abstraction boundary are inextricably linked. *Cf.* Crootof & Ard, *supra* note 49, at 387–95 (noting the interconnectedness between choice of analogy and legally salient details).

[94] *See* Larry Downes, *Aereo TV: Barely Legal By Design*, HARV. BUS. REV. (March 07, 2013), https://hbr.org/2013/03/aereo-tv-barely-legal-by-desig.

[95] WNET, Thirteen v. Aereo, Inc., 712 F.3d 676 (2d Cir. 2013), *rev'd sub nom.* Am. Broad. Cos., Inc. v. Aereo, 573 U.S. 431 (2014); Fox Television Stations, Inc. v. FilmOn X LLC, 966 F. Supp. 2d 30, 34 (D.D.C. 2013); Fox Television Stations, Inc. v. BarryDriller Content Sys., PLC, 915 F. Supp. 2d 1138 (C.D. Cal. 2012).

[96] FilmOn X, 966 F. Supp. 2d at 34.

[97] Fox Television Stations, Inc. v. BarryDriller Content Sys., *PLC*, 915 F. Supp. 2d 1138, 1146 (C.D. Cal. 2012).

One complicating element in these cases was a comparison between these new services and cable providers. Broadcast companies make a lot of money by licensing their content to cable companies—content that is otherwise broadcast free over the air. As the Court explained in *Aereo*, when Congress updated the Copyright Act in 1976, a primary reason was to overturn a case holding that cable TV providers were not publicly performing, and thus did not need to license the content to show it.[98] Thus, after 1976, the Copyright Act clearly meant that a cable company retransmitting broadcast signals to many customers at once without a license would be infringing. Thus, when the networks sued each of these companies, Aereo, BarryDriller, and FilmOn X, the networks argued that what the upstart companies were doing was functionally equivalent to a cable company retransmitting illegally. Thus, the legally relevant question about Aereo's technology was whether the retransmission occurred in a mass transmission, available to anyone who tuned in, or solely existed on an individual antenna by a customer's request.

The cases turned into battles of analogies.[99] The networks argued that Aereo was similar to a cable provider, just with a funny looking circuit board, and Aereo argued that it was doing the equivalent of selling miniature rooftop antennas with exceptionally long cords that reached all the way to Aereo's servers. To look at it differently, this was also a battle over the proper abstraction boundary. To the plaintiffs, the proper abstraction boundary surrounds the entire board, with all 10,000 antennas. From that perspective, it looks like cable retransmission: broadcast signals come in, thousands of retransmitted signals go out to all their customers, just like cable. To the defendants, the proper boundary of the system was the single antenna: broadcast signal in, then out to a single viewing device when requested. From that perspective, it matters no more that there are 10,000 antennas on a circuit board than that millions of homes have antennas on their rooftops. The system is defined as just the one antenna, and so what if there are many of them active at once?

When the case got to the Supreme Court, the plaintiffs won. The Court held that "these behind-the-scenes technological differences do not

[98] Am. Broad. Cos., Inc. v. Aereo, 573 U.S. 431, 432 (2014).

[99] *See* Mark P. McKenna, *The Limits of the Supreme Court's Technological Analogies*, SLATE (June 26, 2014), https://slate.com/technology/2014/06/abc-v-aereo-ruling-the-supreme-courts-terrible-technological-analogies.html.

distinguish Aereo's system from cable systems, which do perform publicly."[100] The dissent, for its part, characterized the majority opinion as "guilt by resemblance" to cable,[101] judged by a vague totality of the circumstances test.[102] The dissenting justices would have looked more closely at the volitional conduct at issue. Though they do not explicitly refer to how they would characterize the technologies, the opinion generally echoes the court below, which observed that "*Cablevision* . . . held that technical architecture matters."[103]

On both sides, *Aereo* was a debate about what copyright law says is the right way to view technology. But an examination of both the Supreme Court case and cases below reveals a subtle, but real difference in approach. The Supreme Court majority did not examine the normative claims embedded in the technology and how it could have been designed differently, but rather rejected any analysis of the technological design in principle.[104] The Court's choice to refer to the technological differences between cable and the systems at issue as "behind the scenes" demonstrates that rather than delve into the technology and decide what version would best comport with the law, they simply refused to examine the technology at all.[105] Even though both parties denied that Aereo's system was, in important respects, a cable system, the Court's ruling suggests, as Mark McKenna put it, that it just "could not shake

[100] *Aereo*, 573 U.S. at 433.

[101] *Id.* at 457 (Scalia, J., dissenting)

[102] *Id.* at 461.

[103] WNET, Thirteen v. Aereo, Inc., 712 F.3d 676, 694 (2d Cir. 2013).

[104] *See Aereo*, 573 U.S. at 442 (stating that Aereo is similar to cable because it is a "commercial enterprise[] whose basic retransmission operations are based on the carriage of copyrighted program material" (quoting H.R. Rep. No. 94–1476, pp. 89 (1976)); *id.* at 446 ("Viewed in terms of Congress' regulatory objectives, why should any of these technological differences matter? They concern the behind-the-scenes way in which Aereo delivers television programming to its viewers' screens."). The Court did reason about one aspect of the technology—that the user initiates the transmission, *id.* at 443—but that would be true of either of the abstractions at issue, so it not relevant to the choice.

[105] *See* Yvette Joy Liebesman, *When Does Copyright Law Require Technology Blindness? Aiken Meets* Aereo, 30 BERKELEY TECH. L.J. 1383, 1423 (2015) (arguing that *Aereo* endorsed "technology blindness" as a method). It's a close call whether to view the Court's decision to call the technological details "behind the scenes" indicates that they refused to engage the technology, or that they chose the abstraction favored by the plaintiffs. The Court's dismissive language suggests the former, but the latter could be another valid interpretation.

the sense that it is very much *like* a cable system," and ruled on that basis.[106] This posture of intentional ignorance was unnecessary to the outcome, however. Lower courts had no trouble coming to the same conclusion while analyzing the technological design. In *Fox Television Stations v. BarryDriller*, for example, the district court still found for the plaintiffs partly because the lack of proximity to the transmission made these systems meaningfully less like individual transmissions and more like a broadcast to many people at once.[107] This was a ruling that engaged with the design choices, and explained why a proper legal understanding of the "system" was one scoped with all of its outputs rather than the one sent to each individual user. In fact, in the retransmission cases, every judge at every level, *except* the justices in the eventual Supreme Court majority, understood that they needed to incorporate an analysis of technological design choices.

The majority's refusal to engage with the technology does not absolve them of the responsibility for that choice—it merely makes the choice unjustified. As the courts below split on outcome, such analyses did not actually dictate the outcome of any case, and the Court could have come to the same conclusion in a more informed manner. In later cases, *Aereo* has unfortunately come to represent the idea that in copyright, the relevant technological perspective is that of the end user, a perspective that hides the technical details and treats the technology as a black box,[108] but it did not have to be that way.

[106] McKenna, *supra* note 99; *See Aereo*, 573 U.S. at 451 (majority opinion) ("In sum, having considered the details of Aereo's practices, we find them highly similar to those of the CATV systems.")

[107] Fox Television Stations, Inc. v. BarryDriller Content Sys., PLC, 915 F. Supp. 2d 1138, 1146 (C.D. Cal. 2012)

[108] *See, e.g.*, Goldman v. Breitbart News Network, LLC, 302 F. Supp. 3d 585, 595 (S.D.N.Y. 2018) ("[T]his Court reads *Aereo*, while not directly on point, as strongly supporting plaintiff's argument that liability should not hinge on invisible, technical processes imperceptible to the viewer."); Malibu Media, LLC v. Bui, No. 1:13-CV-162, 2014 WL 12469955, at *1 (W.D. Mich. July 21, 2014) (holding that the technological distinctions between the instant case and technology held to be infringing in precedent "make[] no difference" because the technology is "functionally indistinguishable from the perspective of both the copyright holder and the ultimate consumer of the infringed work.");

C. Copyright's Functionality Doctrines

For our final example, we turn to functionality doctrines in copyright law. An important question for all intellectual property regimes is whether an aspect of a product is functional or not.[109] Copyright, trademark, and design patent law all seek to accomplish different policy goals. Copyright aims to protect expression, trademark to prevent consumer confusion, and design patents aim to protect the ornamental features of useful products. None of these aims to protect the functional aspects of a product. Utility patents are the type of intellectual property that protects functionality. Because monopolies on functional products are seen as a greater burden on the public than nonfunctional products, the tradeoff for this protection is that utility patents have more requirements and shorter terms than other forms of IP protection.[110] It is not always easy to screen out the functional from the expressive, the distinctive, or the ornamental, however. The base of a lamp might also be a sculpture[111] or the dual spring nature of a road sign might both mark it as the distinctive style of a single manufacturer and help it resist high winds.[112] As a result, each area of IP has created different doctrines in place to screen out functional elements.[113] In this section, because we present this argument merely as an illustration, we will only discuss copyright, first examining nonliteral copying of software and then turning to copyright's "useful articles" doctrine.

1. Non-Literal Copying of Software

Software is composed of written words that perform a function. This has always been an awkward fit for copyright law, as it protects expression, not function.[114] In 1980, Congress updated the Copyright Act to clarify that

[109] *See generally* Christopher Buccafusco & Mark A. Lemley, *Functionality Screens*, 103 VA. L. REV. 1293 (2017).

[110] *Id.*

[111] Mazer v. Stein, 347 U.S. 201, 202 (1954).

[112] TrafFix Devices, Inc. v. Mktg. Displays, Inc., 532 U.S. 23, 32 (2001)

[113] *Id.*

[114] Google LLC v. Oracle Am., Inc., 141 S. Ct. 1183, 1208 (2021) ("The fact that computer programs are primarily functional makes it difficult to apply traditional copyright concepts in that technological world."); Pamela Samuelson, *Staking the Boundaries of Software*

computer programs are protectable content.[115] Thus, to the extent the written code itself is creative expression, copyright protects it. But there are two limitations on the degree to which code is protectable. The first is copyright's idea-expression dichotomy. Only the original expressions of ideas are protectable, not the ideas themselves, and because code is primarily functional, much of it constitutes an implementation of an idea.[116] The second limitation is that the Copyright Act excludes protection for "procedures, processes, systems, and methods of operation,"[117] which has led to holdings that "algorithms, mathematical constants, rules editing methods, methods of calculation, command structures, data structures, interfaces necessary to interoperability, and functional program behavior" are not protectable.[118]

Because code is functional, it is often copied as a matter of practice. As in legal practice, where language that is known to function a certain way is copied from brief banks and old contracts, programmers will copy code wholesale into a new environment to avoid both reinventing and retesting the wheel.[119] For example, the recent Supreme Court case *Google v. Oracle* involved Google's copying the Java application programming interface

Copyrights in the Shadow of Patents, 71 FLA. L. REV. 243, 247 (2019) (*Oracle* is the latest exemplar of the vexing conceptual difficulties that computer programs have posed for both copyright and patent laws over the past fifty-some years [because intellectual property] regimes that have historically assumed that a particular creation was either a writing or a machine, but could not be both at the same time.)

[115] 17 U.S.C. 101 as amended by Pub. L. 96-517, Sec. 10(a), 94 Stat. 3028. (1980).

[116] Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 703 (2d Cir. 1992).

[117] 17 U.S.C. § 102 (b)

[118] Pamela Samuelson, *Functionality and Expression in Computer Programs: Refining the Tests for Software Copyright Infringement*, 31 BERKELEY TECH. L.J. 1215, 1239–40 (2016) ("Courts in software copyright cases have identified several types of nonliteral elements of programs as unprotectable procedures, processes, systems, or methods of operation under § 102(b). Among the elements that courts have filtered out are algorithms, mathematical constants, rules editing methods, methods of calculation, command structures, data structures, interfaces necessary to interoperability, and functional program behavior.) (footnotes omitted).

[119] *See* Google LLC v. Oracle Am., Inc., 141 S. Ct. 1183 (2021) ("It would be difficult, perhaps impossible, for a programmer to create complex software programs without drawing on prewritten task-implementing programs to execute discrete tasks.")

(API),[120] owned by Oracle, in order to allow programmers who had already learned Java to build within the Android environment (Google's mobile operating system).[121] The questions presented to the Court concerned both the copyrightability of the Java API and whether the copying was fair use.[122] Though the Court had the opportunity to rule on the limits of code's protection, it opted instead to avoid a bigger shakeup of the software industry and instead issued a narrow, fact-intensive holding that, assuming the code was protectable, Google's copying was fair use.[123]

Though limits on direct copying of software are important, the more interesting set of cases for our purposes are the instances of nonliteral copying, "copying that is paraphrased or loosely paraphrased rather than word for word."[124] In infringement cases that allege nonliteral copying, both with respect to software and otherwise, courts look for substantial similarity between the original work and the allegedly infringing copy.[125] But whereas in a case involving art or music, a court might look at the whole, comparing the "overall look and feel" of the pieces,[126] courts take a different approach to nonliteral copying of code that requires breaking the piece—the written computer program—into its constituent elements. To do this, courts most commonly use the "abstraction-filtration-comparison" test,[127] originating in *Computer Associates, Inc v. Altai*.[128]

As the name suggests, the test has three parts. First is "abstraction," a process by which "a court should dissect the allegedly copied program's structure and isolate each level of abstraction contained within it."[129] The purpose of the abstraction step is to identify those aspects of the work that are ineligible for copyright protection so that they can be filtered out in the second

[120] An API is a type of software interface that allows computer programs to communicate with each other, while hiding the irrelevant details of each program from the other. Standardized APIs allow for greater interoperability between programs. *See API*, Wikipedia https://en.wikipedia.org/wiki/API.

[121] *Google*, 141 S. Ct. at 1193.

[122] *Id.* at 1197.

[123] *Id.*

[124] Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 814 (1st Cir. 1995)

[125] *See, e.g.*, Blehm v. Jacobs, 702 F.3d 1193, 1199 (10th Cir. 2012).

[126] *See, e.g.*, Boisson v. Banian, Ltd, 273 F.3d 262, 266 (2d Cir. 2001).

[127] *See* Mark A. Lemley, *Convergence in the Law of Software Copyright?*, 10 HIGH TECH. L.J. 1, 2 (1995) (noting this is the most common test); Samuelson, *supra* note 118, at 1230–37 (discussing the test at length).

[128] Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 706–11 (2d Cir. 1992).

[129] *Id.* at 707.

step.[130] This idea of abstraction is not *exactly* the formal programming method we discuss throughout this paper, though the application is closely related. Stemming from case law about novels and screenplays, the idea of abstraction in copyright law is to describe the work in successive levels of generality, in order to separate the core ideas from the particular expression. So if a play contain certain tropes or character types, those are more general ideas than the dialogue in a scene or specific characters. Though not the exact same concept, the application here to computer programs is not coincidental. In *Altai*, the court discussed the work of a programmer as "moving from the general to specific," breaking down the problem into smaller tasks[131]—and those smaller tasks, those subroutines are the implementation of the formal abstractions that programmers create. Thus, when discussing the possibly of copyright protection for a program and its subroutines,[132] the court is asking whether there is an element of creative expression in the ways that a programmer implemented the abstractions in code.

The second step of the process is filtration. Code could be unprotected for a number of reasons: 1) the code was written a certain way to effect functional concerns such as efficiency, portability, or compatibility,[133] 2) the programmers' design choices were "circumscribed by extrinsic considerations" such as the limits of the machine it runs on, industry demands, or standard programming practices,[134] or 3) the code draws on the public domain, copying either public domain software,[135] or "commonplace programming techniques, ideas, and know-how."[136] In the filtration step, a court filters out from its consideration any aspects of the code that are

[130] Country Kids 'N City Slicks, Inc. v. Sheen, 77 F.3d 1280, 1284–85 (10th Cir. 1996) ("At the abstraction step, we separate the ideas (and basic utilitarian functions), which are not protectable, from the particular expression of the work."), *quoted in* Paycom Payroll, LLC v. Richison, 758 F.3d 1198, 1205 (10th Cir. 2014) (applying the principle to software).

[131] *Id.* at 697; *see also* Gates Rubber Co. v. Bando Chem. Indus., Ltd., 9 F.3d 823, 834 (10th Cir. 1993) ("The abstractions test is especially well suited to the dissection of computer programs because the test breaks down a program in a way that parallels the typical development of a program.")

[132] *Altai,* 982 F.2d at 697, 707.

[133] *See Altai,* 982 F.2d at 707; Samuelson, *supra* note 118, at 1221 ("Courts have deemed the functional requirements for achieving compatibility to be unprotectable elements of these copyrighted programs, even though more than a modicum of creativity may have imparted originality to these elements.")

[134] *Id.* at 709–10.

[135] *Id.* at 710.

[136] Samuelson, *supra* note 118, at 1231.

unprotected. Importantly, copyright's merger doctrine also plays out in the filtration step.[137] It states that where there is both expression and functionality, if the expression is the only way or one of only several ways to achieve that functionality, the expression and function merge, leaving the whole unprotected.[138] Finally, the third step is comparison. If after the filtration, the court is left with a "kernel" of creative expression, it compares that kernel to the original work to determine whether it is substantially similar.[139]

This test asks courts to do for copyright something parallel to what we are recommending courts do more generally. The *Altai* court in fact described the task as a type of "reverse engineering," noting that to accomplish it, a court must "essentially [] retrace and map each of the designer's steps—in the opposite order in which they were taken during the program's creation."[140] In order to determine whether there is anything to filter out, a court must look at the abstractions that developers created and how they were implemented in code, consider how many alternate possibilities exist and what they are, and consider the reasons that programmers implemented the code in the specific ways they did. It is not *precisely* the same task as we argue for, as in copyright the court's subject of interest is the words that implement the abstractions rather than the abstraction choices themselves, but nonetheless, the doctrine requires courts to unpack the abstraction choices, understand what alternatives exist, and for what reasons the choices were made.

2. Useful Articles

The second copyright doctrine of interest is about designs of "useful articles" such as clothing, cars, electronics, and furniture. Such designs themselves are protectable only if they "incorporate[] pictorial, graphic, or sculptural features that can be identified separately from, and are capable of existing independently of, the utilitarian aspects of the article."[141] So where a belt buckle is itself a work of art, it is protectable because the artistic aspects do not change the utility of the buckle, which would hold pants up with or

[137] *Altai,* 982 F.2d at 708.
[138] *Id.*
[139] *Id.* at 706.
[140] *Id.* at 707.
[141] 17 U.S.C. § 101 (definition of "[p]ictorial, graphic, and sculptural works")

without the artistic embellishment.[142] But where a mannequin's only distinctive feature is that it is human torso shaped, that choice of design is intrinsic to the function of displaying clothing, and will not be protected.[143]

As with software, determining how to filter out the expressive parts of a functional design is a conceptually difficult task. The challenge is how to conceptually separate the artistic in the design from the functional. As Rebecca Tushnet has put it, the distinction can be seen as a case of a "design on" a useful article versus a "design of" one.[144] Courts have for a long time struggled to define a single test for conceptual separability.[145] Some courts have asked whether the design is "primary" and the utilitarian function "subsidiary."[146] Others have held the design separable only if it is not "necessary" to the utilitarian function.[147] Still other courts asked if the design "reflect[s] the designer's artistic judgment exercised independently of functional influences."[148] These are not the only tests either.[149] The proper test itself is less important for our purposes than the point that once again, courts have historically been required to unpack the technology and determine its purpose and possible alternatives. It's not possible to determine whether a design or function is "primary" or "subsidary", whether a design is "necessary" or not, or whether it reflects artistic judgment independent of function without a deep understanding of function and the role that the design choices play in it.

In 2017, the Supreme Court decided *Star Athletica v. Varsity Brands* "to resolve [the] widespread disagreement over the proper test."[150] Unfortunately, according to copyright scholars, the Court instead threw out years of precedent, misread the statute, and undermined the little sense that

[142] Kieselstein-Cord v. Accessories by Pearl, Inc., 632 F.2d 989, 990 (2d Cir. 1980)

[143] Carol Barnhart Inc. v. Econ. Cover Corp., 773 F.2d 411 (2d Cir. 1985).

[144] Rebecca Tushnet, *Shoveling A Path After Star Athletica*, 66 UCLA L. Rev. 1216, 1237 (2019).

[145] Daan G. Erikson, *Copyright Protection for Conceptually Separable Artistic Features Post-Star Athletica: A Useful Article on Useful Articles*, 18 CHI.-KENT J. INTELL. PROP. 56, 61–62 (2019) ("[D]ifferent circuit courts and scholars used at least nine different tests, with courts in the same circuit sometimes even diverging, as courts struggled with the best language to protect creators of various useful articles that featured decorative elements.")

[146] *Kieselstein-Cord*, 632 F.2d at 993.

[147] *Carol Barnhart*, 773 F.2d at 419.

[148] Brandir Int'l, Inc. v. Cascade Pac. Lumber Co., 834 F.2d 1142, 1145 (2d Cir. 1987).

[149] *See* Varsity Brands, Inc. v. Star Athletica, LLC, 799 F.3d 468, 484–85 (6th Cir. 2015) (describing nine different tests).

[150] Star Athletica, L.L.C. v. Varsity Brands, Inc., 580 U.S. 405, 409 (2017)

the doctrine previously made.[151] The case involved a copyright for the design of a cheerleading uniform, including an arrangement of lines and chevrons.[152] The court held that it was protectable because it "can be perceived as a two- or three-dimensional work of art separate from the useful article" and would qualify as a pictoral, graphic, or sculptural work if "imagined separately from the useful article into which it is incorporated."[153] Before *Star Athletica*, where functional and expressive elements overlapped in useful articles, they were held to be unprotected, but to the extent the doctrine is clear now, the Court flipped the test such that only features that are purely functional are unprotected.[154]

Thus, the Supreme Court again created a new test that actually does not require courts to examine technology. Instead, it seems that if there is anything artistic, the functionality takes a back seat. Nonetheless, much like with *Aereo*, the work in the lower courts using the prior precedent demonstrates that where the law clearly requires a court to unpack technological design choices, there is nothing unusual about courts doing so.

III. DECONSTRUCTING DESIGN IN CASES NOT ABOUT TECHNOLOGY

The prior Parts have shown why courts should deconstruct technology, and that courts are able to do so. In this Part, we show how such deconstruction can work in cases that are not explicitly about the technology. We focus on the implications of hidden design choices in machine learning in hypothetical cases involving negligence, anti-discrimination law, and criminal

[151] Tushnet, *supra* note 144, at, 1218 ("[T]he case definitively announc[ed] . . . that decades' worth of lower courts' attempts to interpret the copyright statute's treatment of 'useful articles' were worthless. . . . [T]he majority opinion has internal tensions that threaten to tear it apart . . . . The best I can say about the case is that my copyright students' situation has improved: Instead of having to learn up to nine incompatible approaches to copyright protection for useful articles, as in the past, they only have to learn one that's self-contradictory."); Barton Beebe, *Star Athletica and the Problem of Panaestheticism*, 9 UC IRVINE L. REV. 275, 290 (2019) ("[I]n the wake of *Star Athletica* . . . courts may revive . . . a new chaos of approaches not unlike what preceded *Star Athletica*."); Lili Levi, *The New Separability*, 20 VAND. J. ENT. & TECH. L. 709, 729 (2018) ("Having overturned prior law, the Court offered a conceptual new test. . . . Each of the two prongs of the new test is problematically indeterminate.")

[152] *Star Athletica*, 580 U.S. at 411.

[153] *Id.* at 409.

[154] Buccafusco & Lemley, *supra* note 109, at 1335.

justice. The discussion aims to demonstrate how abstraction choices compete with legal and factual determinations that belong to the courts. We have chosen these areas of law both because they are relevant to existing machine learning applications and because the import of the abstraction choices means different things for the parties in each, allowing for a more varied discussion.

A. Negligence

As we discussed in Part I, an abstraction boundary sets the expectations of responsibility for both those inside and outside the boundary. Where there is a well-defined boundary, if the developer *is* responsible for an outcome, the user is not, and vice versa. If the designer of the coffeemaker is not responsible for the grind (and therefore does not provide a built-in grinder), the user knows that she is. That choice of abstraction boundary is a well-defined choice. While the abstraction boundary evinces an outer limit to what product developers should have to consider or model about the outside world, it also instructs those on the outside of the boundary—the users—what they need to know about its internal operations in order to interact with the product.

A tort case is concerned with both sides of the boundary. Consider a lawsuit that results from a physician's use of computer vision software developed with deep learning to assist in cancer diagnosis. Suppose that the software misclassifies a tumor as unlikely to be cancerous, and the diagnosing physician follows this recommendation, causing the cancer to remain untreated for some amount of time, eventually killing the patient. The patient (through her estate) could sue the doctor or hospital for negligence. They could also sue the developers of the software under a products liability theory. The judge and jury will be tasked with deciding which of these defendants, if any, is liable for the injury.

We want to offer a few caveats before proceeding with this example. First, a medical malpractice hypothetical may not be an ideal example for illustration because the negligence analysis is complicated by the professional malpractice standard.[155] In medical malpractice cases, the professional standard of care is determined by experts rather than the judgment of lay juries. This perhaps renders our claim that courts should analyze abstractions futile because medical experts will ultimately define reasonable caution,

[155] *See, e.g.*, Sheeley v. Memorial Hospital, 710 A.2d 161 (R.I. 1998).

rather than the court.[156] We nonetheless proceed here with a medical malpractice hypothetical because it is the most realistic and urgent. Machine learning diagnosis tools are already in use, and not many other clean examples of ML-driven decision-assistance technology are.[157] At the same time, states are moving away from the traditional strict reliance on custom in medical malpractice.[158] More importantly, though, the analysis is general enough that it would apply in other negligence-related AI fact patterns, such as semi-autonomous driving, data security, and investment advice.[159]

Two other caveats are in order. To focus on the negligence claim, we set up a hypothetical in which the manufacturer is not sued, so that the negligence claim stands alone. In reality, if there is any question about division of responsibility, the manufacturer would be joined in the lawsuit, but we want to demonstrate the how abstraction choices relate to the negligence claim independent of the manufacturer's potential liability for a defective product. Finally, in this discussion, we ignore the historical reluctance of courts to find tort liability for software cases because it is largely inapplicable in this context with tangible harms.[160]

---

[156] *Id.*

[157] Jane R. Bambauer, *Dr. Robot*, 51 U.C. DAVIS L. REV. 383, 387 (2017) (discussing 23andMe and IBM's Watson for Oncology project as examples of AI medicine); Jeffrey De Fauw et al., *Clinically Applicable Deep Learning for Diagnosis and Referral in Retinal Disease*, 24 NATURE MED. 1342, 1342-50 (2018); W. Nicholson Price II, *Black- Box Medicine*, 28 HARV. J.L. & TECH. 419, 426 (2015); W. Nicholson Price II, *Regulating Black-Box Medicine*, 116 MICH. L. REV. 421, 425-26 (2017); Stephen F. Weng, Jenna Reps, Joe Kai, Jonathan M. Garibaldi & Nadeem Qureshi, *Can Machine-Learning Improve Cardiovascular Risk Prediction Using Routine Clinical Data?*, PLOS ONE, April 2017, at 1, 1; Katie Chockley & Ezekiel Emanuel, *The End of Radiology? Three Threats to the Future Practice of Radiology*, 13 J. AM. COLL. RADIOL. 1415 (2016).

[158] Michael D. Greenberg, *Medical Malpractice and New Devices: Defining an Elusive Standard of Care*, 19 HEALTH MATRIX 423, 428–29 (2009) ("Although customary practice remains the legal standard for malpractice in many states, almost half of the states have adopted an objective "reasonable care" standard instead.")

[159] *See* Andrew D. Selbst, *Negligence and AI's Human Users,* 100 B.U. L. REV. 1315, 1329–30 (2020).

[160] Courts have historically been reluctant to find tort liability for software crashes. Bryan Choi describes three main reasons for this reluctance: the intangibility or purely financial nature of harms from software crashes, protectionism of the then-nascent software industry, and complexity of software that suggests that makes it "seemingly impossible to identify marginal-cost measures that can or should be taken to improve software safety." Bryan H.

The negligence claim will boil down to a question of whether the doctor breached his duty of care by unreasonably relying on the software's recommendation. How would a court go about asking that question? We can start off by positing that the software must be at least accurate enough that the choice to use it at all is not itself unreasonable. If not, then this is an easier case—though perhaps it's the hospital, not the doctor who is to blame. Once we are past that threshold, then, the breach question becomes a question of what it was reasonable for a doctor to know about how the software works, and where it is reasonable to rely on its recommendations.

In an extreme case, we can imagine a doctor is provided the software, instructed on how to use it, told it is more accurate than doctors on average, and then told nothing else about it.[161] Such an example presents the technical object as a pure take-it-or-leave-it black box. Every possible interaction with context is considered internal to the abstraction, and therefore not the doctor's responsibility. The abstraction states that software works some large percentage of the time, and as long as that remains true, the user need not consider anything else about it. If the court accepts this version of what constitutes the software, it is hard to imagine how a doctor can be held liable for breach, other than in extremely obvious cases of error, such as where the machine misses a tumor visible to the naked eye.[162] The doctor would be told her instincts and training are less reliable than the machine, and have nothing else to go on, so would have to accept the machine's result—a court holding this to be negligence would have to hold all use of the AI to be negligent, and we posit above that this is not the case. Because this example is so extreme, a court may not simply accept it unquestioningly, but even this simple example shows that the abstraction—how the designers of the technology defined its boundaries—can dictate what the user of the device can be reasonably expected to know.

Now consider a more plausible case. Perhaps the tumor detection algorithm's generalization performance was estimated to be high, but certain

Choi, *Crashworthy Code*, 94 Wash. L. Rev. 39, 64 (2019). The first two rationales just don't apply to software in a medical context, and just as Choi points out with respect to autonomous vehicles, there is a world of difference between the point that software will always contain bugs, and the idea that no errors are foreseeable. The foreseeability of some of these errors is a primary reason that courts must unpack technology.

[161] A. Michael Froomkin et. al., *When AIs Outperform Doctors: Confronting the Challenges of a Tort-Induced Over-Reliance on Machine Learning*, 61 Ariz. L. Rev. 33, 50 (2019) (arguing that when AI diagnosis is simply superior to humans its use will become the standard of care).

[162] Selbst, *supra* note 159, at 1339.

conditions caused it to behave worse than expected. For instance, the training data used to develop the model and estimate generalization performance contained uniformly-lit images, but the doctor or hospital tried to use the software on an image taken with a bright light somewhere in the room. In general, if a supervised model is deployed on data drawn from the same statistical distribution as the validation data, generalization error estimates will be similar in deployment; if it is not, the best model for the test data may be based on spurious or unhelpful patterns which do not translate to the real world.[163]

Consider the abstraction here. Recall that an abstraction is composed of well-defined inputs and outputs and the function that transforms inputs to outputs. The input definition is therefore part of the abstraction. The inputs could have been defined in two ways: Either the model took in "images" or "uniformly lit images." If the input was defined as the latter, that implies that a non-uniformly lit image is not an acceptable input to the model, meaning that the lighting is the user's responsibility. If the input is defined as just any old image, then that suggests that the technology should work on any image, and it is the developer's responsibility to either train on more varied light or have software that automatically compensates for the lighting before processing.

By the time a court encounters the injured patient, the technology has been set; either the proper input was "images" or "uniformly lit images." The abstraction choice was made long ago for reasons related to engineering or business. Perhaps it is better for the company's sales to say their technology can handle images of differing lighting quality—or more likely, not say anything at all. Or perhaps, if it's set up the other way, the company only takes only uniformly images as the input because it was cheaper for them to offload the responsibility of proper lighting to the medical professionals than to write software that can handle myriad lighting conditions. Depending on how the input was defined, it will communicate to the user whether they need to care about specific lighting.

Suppose that in this lawsuit, the developers did not specify lighting conditions, and the doctor argues that she used the machine properly and

[163] Rich Caruana, et al., *Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-day Readmission*, Proc. 21st ACM SIGKDD INT'L CONF. ON KNOWLEDGE DISCOVERY & DATA MINING (2015); Martin Arjovsky et al., *Invariant Risk Minimization* (2019); Andrew Ilyas et al., *Adversarial Examples Are Not Bugs, They Are Features* Proc. Advances in Neural Information Processing Systems (2019).

that errors caused by the lighting issues were unforeseeable to her. Here, the court should question what work abstraction boundaries are doing. In this case, the doctor would argue that because the technology accepted all images, there was no way for her to know that her images would cause a misdiagnosis. If a doctor uses a machine that takes images of tumors as an input and outputs a probabilistic tumor diagnosis, the doctor would have no reason to suspect that lightning in background mattered so much. The design of technology communicated that lighting is internal to the abstraction, which means that the doctor should not have to know about challenges with lighting issues. Note that we are not speaking about the developers' intent either. They need not have intentionally designed the technology to internalize the lighting conditions—given the facts of the hypothetical, it was more likely an oversight—but nonetheless the design of the technology communicates a position that relieves the doctor of responsibility.

Perhaps this is the right result. If the developers got it wrong, but the doctor had no reason to suspect the tool was compromised and used it correctly according to its specifications, then the doctor likely acted reasonably under the circumstances, which is all that is required in a negligence case. Perhaps the lighting issue was unforeseeable for either the developers or the doctor and thus neither party should be legally at fault. Our point is not that a court's understanding of technology will always change the result, just that the particular arrangement of the technological elements as they exist points to certain outcomes over others, and that the court should understand and critique the arguments implied by the design rather than accept them without noticing or questioning. In this scenario, perhaps a court would find that these technologies are new enough or known to be finicky enough that it might be safer to hold medical professionals responsible irrespective of the apparent functionality of the device. Perhaps AI errors in general are common enough that errors should be deemed foreseeable even if this particular error could not have been named specifically ahead of time.[164]

Presumably, even in a case with no products liability claim, the court should want to hear expert testimony on the creation of the device and the

[164] *See* W. Jonathan Cardi, *Reconstructing Foreseeability*, 46 B.C. L. REV. 921, 939 (2005) (discussing the need for judgment calls in determining the correct level of generality for a foreseeability determination); David G. Owen, *Figuring Foreseeability*, 44 WAKE FOREST L. REV. 1277, 1277–79 (2009) (describing foreseeability as policy judgments about where liability should end rather than factual determinations).

expectations of the users. In this case, an expert could testify about what is reasonable within the computer vision field. The court could then ask what the doctor—or hospital that purchased the software—should have to know about the inner workings of the technology to satisfy the legal goals of tort law. Should they have to run their own tests? Or at least check that use of the tool in general has been peer reviewed? We are not going to expect the doctor or hospital to be machine learning experts in addition to medical experts. (This is why products liability exists—so that we can trust the basic functionality of products.) But perhaps the cost to hospitals of taking better photos is lower than the cost to the developers of covering every potential lighting condition, such the medical professionals become the least cost avoiders. Ultimately, these are questions that belong to courts; legal reasoning should dictate where responsibility lay, and that includes questioning the claims the technology makes.

B. Discrimination

The most commonly cited AI harms relate to bias, fairness, or discrimination. By now it is well understood that choices in problem definition, training data, or other aspects of the machine learning problem can result in differential outcomes on protected classes.[165] But there is disagreement about whether anti-discrimination law can address these discriminatory harms.[166] One element of that disagreement is the extent to which a discrimination defendant is entitled to buy a system off the shelf and use the black-box nature of the item as a shield against liability. If the discrimination defendant is responsible to look under the hood, then they can be held liable, but if not then it is much more difficult to hold them responsible. We believe that this can be understood as a problem of abstraction boundaries.

In the United States, discrimination law canonically has two halves. "Disparate treatment" is generally taken to mean intentional discrimination—though could also encompass unintentional differential treatment[167]—and "disparate impact" doctrine is purportedly effects-

[165] Barocas & Selbst, *supra* note 51, at 693; Kim, *supra* note 51.

[166] Barocas & Selbst, *supra* note 51; Kim, *supra* note 51; Deborah Hellman, *Measuring Algorithmic Fairness*, 106 VA. L. REV. 811 (2020); Ifeoma Ajunwa, *The Paradox of Automation as Anti-Bias Intervention*, 41 CARDOZO L. REV. 1671 (2020).

[167] *See* Katie Eyer, *The But-for Theory of Anti-Discrimination Law*, 107 Va. L. Rev. 1621, 1631 (2021).

based,[168] though perhaps more accurately described as effects-triggered. Our concern here is primarily with disparate impact, because that is the kind of discrimination claim most likely to be generated by the use of machine learning.[169]

Disparate impact today involves a three-step burden-shifting scheme.[170] First, a plaintiff must point to an action the defendant took that had a disproportionate impact on members of a protected class.[171] Next, the defendant can respond by arguing that it is a "business necessity."[172] The business necessity test is the heart of disparate impact doctrine. It asks whether the disproportionate impact is justified by some legitimate business purpose.[173] In employment, the standard is generally a low one: whether the challenged practice is job-related.[174] The third step in the analysis allows a plaintiff to demonstrate that there was a less discriminatory alternative that the defendant refused to use.[175]

The case of algorithmic discrimination arises where a person—say an employer—must make an allocative decision such as promotion or hiring and decides to use a machine learning tool to predict the best outcome among candidates. The tool is advertised as the best available predictor of future job outcomes among candidates. Thus, when it evinces a disparate impact, triggering the three-part test, the defendant has a relativity easy time satisfying the business necessity defense—choosing a tool that would give him the best hiring outcome is surely job-related.[176] The last part of the test, however, asks whether there was a less discriminatory alternative to accomplish what the employer wanted. This is where the abstraction boundary comes in. The easiest way to imagine a similarly successful, but less discriminatory tool,

[168] *See* Texas Dep't of Hous. & Cmty. Affairs v. Inclusive Cmtys. Project, Inc., 576 U.S. 519, 533 (2015).

[169] Barocas & Selbst, *supra* note 51, at 693.

[170] 42 U.S.C. § 2000e-2(k)(1)(A).

[171] *Id.*

[172] *Id.*

[173] Texas Dep't of Hous. & Cmty. Affairs v. Inclusive Cmtys. Project, Inc., 576 U.S. 519, 521 (2015) (decisions that are "artificial, arbitrary, and unnecessary barriers" violate disparate impact).

[174] Barocas & Selbst, *supra* note 51, at 705 (noting the low standard across all circuit courts).

[175] 42 U.S.C. § 2000e-2(k)(1)(A).

[176] Barocas & Selbst, *supra* note 51, at 705.

would be to fix the predictive model so it is less discriminatory.[177] But if we start from the position that the model is an unchangeable black box, that option is taken off the table.

To understand this more conceptually, it will be helpful to zoom out in our understanding of what anti-discrimination law is trying to accomplish. Anti-discrimination law aims to address specific acts of discrimination taken by a defendant or against a plaintiff.[178] Today, most people recognize that there are forms of institutional and structural discrimination that cannot be traced back to a single action, but instead reflect the effects of compounded historical animus and decades of discriminatory policies.[179] But anti-discrimination law does address every instance of discrimination understood this way; rather, each individual lawsuit addresses only that subset of discriminatory harm fairly attributable to the defendant's conduct, whether intentional or unintentional.[180] This can be seen in the structure of the doctrine. Functionally, while the first step of the three-step disparate impact test is a strict liability, effect-based trigger, the business necessity and less discriminatory alternative tests reintroduce responsibility by asking if the defendant was justified in the action despite the discriminatory effects. But while a defendant is only liable if they are responsible for the discrimination,

[177] There is increasing evidence that the discriminatory aspects of algorithmic models can be improved, often without changing the accuracy of the model, though it is not always easy to do so. *See, e.g.*, Emily Black, Manish Raghavan & Solon Barocas, *Model Multiplicity: Opportunities, Concerns, and Solutions*, PROCEEDINGS OF 2022 ACM CONF. ON FAIRNESS, ACCURACY & TRANSPARENCY 1350 (demonstrating that trained models are often dependent on initial conditions and multiple models can often be trained with the same accuracy); Michael Feldman et al., Certifying and Removing Disparate Impact, PROCEEDINGS OF 21ST ACM SIGKDD INT'L CONF. ON KNOWLEDGE DISCOVERY & DATA MINING 259 (2015).

[178] *See* Alan David Freeman, *Legitimizing Racial Discrimination through Antidiscrimination Law: A Critical Review of Supreme Court Doctrine*, 62 MINN. L. REV. 1049 (1978) (discussing the "perpetrator perspective" of anti-discrimination law); [[others]]

[179] *See* KHIARA M. BRIDGES, CRITICAL RACE THEORY: A PRIMER 148-49 (2019) (defining institutional and/or structural racism as including four elements: 1) lack of intentionality; 2) production by everyday decisions; 3) facial neutrality in decisions; and 4) lack of an identifiable bad actor).

[180] *See* TRISTIN GREEN, DISCRIMINATION LAUNDERING: THE RISE OF ORGANIZATIONAL INNOCENCE AND THE CRISIS OF EQUAL OPPORTUNITY LAW 40–42 (2017) (describing and critiquing the development of an "organizational innocence" paradigm in which courts care at least as much as not unfairly blaming organizations for discriminatory results as it does the discriminatory results themselves).

the lack of liability does not actually mean no harm occurred; rather the discriminatory harm is often what triggered the lawsuit in the first place.[181]

Thus, another way of understanding the function of anti-discrimination law is to determine for an individual instance of discriminatory harm, whether it is the responsibility of the defendant, the responsibility of someone else, or essentially an accident. This sounds a lot like a tort analysis, where we can usually agree there was an injury, and a court must attribute fault.[182] Indeed, Title VII is frequently described as a "statutory tort,"[183] and some consider it a "contemporary extension[] of tort law."[184] While the direct comparison to tort law is controversial,[185] one need not believe that disparate impact law has exactly the structure of a particular tort to understand the latter two steps as examining the question of decisionmaker responsibility for resulting discrimination.[186]

This brings us back to abstractions. When the defendant raises a business necessity defense by saying that he bought a machine learning tool that was advertised as the most accurate solution for his business, he holds up the tool as a complete object; he is relying on the abstraction choices made by developers of the tool. Recall that the very purpose of formal abstraction is to tell a user of a tool that they need not worry about what's going on inside.

---

[181] *See* Noah Zatz, *Disparate Impact and the Unity of Equality Law*, 97 B.U. L. REV. 1357, 1375 (2017) (arguing that we can categorize the discrimination injury as "status causation" and separate that analysis from responsibility for it).

[182] David Oppenheimer famously argued that in operation, despite being seen as divided between intentional and strict liability standards, anti-discrimination law actually has a great deal in common with negligence. David Benjamin Oppenheimer, *Negligent Discrimination*, 141 U. PA. L. REV. 899 (1993); *see also* Richard Thompson Ford, Bias in the Air: Rethinking Employment Discrimination Law, 66 STAN. L. REV. 1381, 1390 (2014) (comparing discrimination to negligence); Sandra F. Sperino, *Rethinking Discrimination Law*, 110 MICH. L. REV. 69, 99 (2011) (referring to the least discriminatory alternative test as negligence).

[183] *See* Charles A. Sullivan, *Tortifying Employment Discrimination*, 92 B.U. L. REV. 1431, 1432 (2012).

[184] John C.P. Goldberg & Benjamin C. Zipursky, *Torts as Wrongs*, 88 TEX. L. REV. 917, 918, 919 (2010); Sandra F. Sperino, *Discrimination Statutes, the Common Law, and Proximate Cause*, 2013 U. ILL. L. REV. 1, 35 (agreeing with Goldberg & Zipursky, *supra*, as a general matter, while arguing that courts should not port tort doctrine wholesale into employment discrimination law).

[185] *See* Sullivan, *supra* note 183, at 1432; W. Jonathan Cardi, *The Role of Negligence Duty Analysis in Employment Discrimination Cases*, 75 OHIO STATE L.J. 1129 (2014); William R. Corbett, *What Is Troubling About the Tortification of Employment Discrimination Law?*, 75 OHIO ST. L.J. 1027, 1032 (2014); Sandra F. Sperino, *The Tort Label*, 66 FLA. L. REV. 1051 (2014).

[186] *See, e.g.*, Zatz, *supra* note 181, at __.

In raising such a defense, the discrimination defendant is relying on this same idea. It's not a trick or bad faith, though. As a practical matter, most employers will not have the technical capacity to build and train such a model themselves. Thus, in most cases they will purchase the software, either off the shelf, as an existing package, or as a bespoke model, co-designed with a technology company. They purchase the technology, which may very well be more predictive of their future job outcomes than other methods they would use to promote employees, and ipso facto, they establish business necessity.

If the court's role is to determine whether the defendant was in fact culpable for the discrimination, then the court's role in the case of algorithmic discrimination is to decide whether a black box defense is good enough. Scholars have proposed different reasons that it should not be. Pauline Kim has argued that discriminatory correlations in a model might be spurious,[187] James Grimmelmann and Daniel Westreich argued that correlations might have no explanation other than relationship to protected class and should be rejected on that basis,[188] and one of us (Selbst) and Solon Barocas have argued that systems that are not based on human-understandable statistical relationships could be rejected unless there is documentation to justify the decisions in building them.[189] There are many reasons to think we can do better than just defer to the availability of the off-the-shelf system. But the lack of technical sophistication of many employers who are increasingly pressured to purchase ready-made technological solutions offers at least some reason to think we should take such a defense seriously.

The question of whether we should permit the black-box defense is therefore open, and once again requires understanding how the boundaries of the abstraction influence the apparent lines of responsibility. Once business necessity is established, the plaintiff bears the burden of proof to demonstrate the existence of an alternative. In the case where companies buy off the shelf, it seems natural to think of the product as a finished good, a take-it-or-leave-it box with the only options being to use it or not. Then the only relevant questions become whether the defendant has the ability to comparison-shop between off-the-shelf AI products with different discriminatory responses. If a plaintiff wants to prove the existence of an alternative, she must show that

[187] Kim, *supra* note 51.

[188] James Grimmelmann & Daniel Westreich, *Incomprehensible Discrimination*, 7 CALIF. L. REV. ONLINE 164 (2017).

[189] Andrew D. Selbst & Solon Barocas, *The Intuitive Appeal of Explainable Machines*, 87 FORDHAM L. REV. 1085 (2018).

there is another piece of software in the market that would have done as well for the company but be less discriminatory; there is not really a way to say that the employer should have changed the model. Then, because is not clear that such a competitor will exist,[190] and courts tend to be deferential to employers regarding what is best for the business,[191] this showing will likely fail. Note the work that the abstraction boundary and choices are doing here. Because the system is off-the-shelf, there is a boundary between the creation of the system and the use of it. It therefore seems natural to suggest that the defendant wasn't in position to prevent the harm.

Now consider an alternative similar to the biased medical imaging model above—suppose the software company offered documentation saying that their software works best on certain specified demographic distributions.[192] This would change the model abstraction—now the model takes as inputs only data that matches the distribution they documented; the requirement to ensure proper use of the tool—that the population tested

[190] Today, at least, comparison shopping is not a realistic possibility. As the market stands now, there are few vendors selling these solutions for each different use cases, leading scholars to warn of the dangers of algorithmic monoculture. *See* Jon Kleinberg & Manish Raghavan, *Algorithmic Monoculture and Social Welfare*, 118 PROC. OF NAT'L ACAD. OF SCIS. 1, 1 (2021) ("The rise of algorithms used to shape societal choices has been accompanied by concerns over *monoculture*—the notion that choices and preferences will become homogeneous in the face of algorithmic curation."); Kathleen Creel & Deborah Hellman, *The Algorithmic Leviathan: Arbitrariness, Fairness, and Opportunity in Algorithmic Decision-Making Systems*, 52 CANADIAN J. PHIL. 26, 27 (2022) (showing how applicants can be arbitrarily excluded from a broad range of opportunities due to algorithmic homogeneity). Even were that not the case, firms neither regularly test for discrimination nor provide public audits, making comparison shopping impossible. *See* Inioluwa Deborah Raji, Andrew Smart, Rebecca N. White, Margaret Mitchell, Timnit Gebru, Jamila Smith-Loud, Daniel Theron & Parker Barnes, *Closing the AI Accountability Gap: Defining an End-to-End Framework for Internal Algorithmic Auditing*, PROC. OF 2020 ACM CONF. ON FAIRNESS, ACCOUNTABILITY & TRANSPARENCY 33 (arguing for the necessity of internal audits before products are released). This may change in the future, but competition is not, at the moment a realistic alternative.

[191] Kim, *supra* note 51, at 908.

[192] This may not appear likely because no software company wants to admit their technology is biased by race or gender, but there is growing recognition within the technical community that increased documentation as to the limits of data or models is important. *See* Timnit Gebru et al., *Datasheets for Datasets*, COMMCN'S ACM, Dec. 2021, at 86, 86; Sarah Holland, Ahmed Hosny, Sarah Newman, Joshua Joseph & Kasia Chmielinski, *The Dataset Nutrition Label: A Framework to Drive Higher Data Quality Standards*, in DATA PROT. & PRIV. 4 (Dara Hallinan et al. eds., 2020); Emily M. Bender & Batya Friedman, *Data Statements for Natural Language Processing: Toward Mitigating System Bias and Enabling Better Science*, 6 TRANSACTIONS ASS'N FOR COMPUTATIONAL LINGUISTICS 587, 588 (2018).

matched the tool's training data—now rests with the user of the tool. Part of Professor Kim's argument above was that employers who use machine learning models should bear the burden to demonstrate that the technology is not overly discriminatory as part of the business necessity test, rather than plaintiffs bearing the burden as part of demonstrating a less discriminatory alternative.[193] Seeing that there are different possible abstraction choices can help support Professor Kim's argument. Recall the facial recognition example in Part I, with three products—amusement park access, bank authentication, and TSA-verification. Let's apply the same reasoning here. Imagine a company called HireZone that has two products: "Los Angeles" and "Salt Lake City." We wouldn't necessarily think of them as the same type of system at all. If a Los Angeles startup said, "we used the Salt Lake City product because it was cheaper and it's basically the same thing," then the court would expect some factual development, perhaps expert evidence, to establish that it is in fact the same. The court would not just assume that one of the parties was entitled to make such a factual assertion. When a company comes in with any type of technology, they are making a factual claim as to what counts as the boundary of that technology—whether it should be a "hiring algorithm" or a "hiring algorithm for a population that looks like Los Angeles." Thus, when a defendant claims that they were helpless because they bought software off the shelf, a court could, where appropriate, assign the burden to them to prove their claim that the boundary of a technology that fails to take demographic context into account is appropriately placed. Perhaps the use of a system design that is known to be flawed for that reason is not in fact a business necessity.

The importance of abstraction principles becomes even clearer by comparing to the scenario where the employer co-develops the software with the technology company. While the end-product may be the same, the lines of responsibility are laid out differently by the design process of the technology. The employer had a hand in some of the internal choices, and therefore the abstraction boundary still points to the employer as responsible—because the designer, unlike the user, is responsible for what happens inside the black box. Accordingly, no employer in this circumstance could plausibly argue that there was no alternative to the technology that resulted by simply pointing to the object. They might still be able to argue it was the best option, but they will have to do more work to establish that fact.

[193] Kim, *supra* note 51, at __ 921.

The liability for such a collaboration would be analogous to product liability's "component parts doctrine."[194] A maker of a component part is generally not liable for a defect in the whole product if their component is not defective, but if they substantially participate in the integration—if it is co-developed essentially—then they become liable as well. The parallel is unsurprising; the very idea of distinguishing between a component and the whole is the same idea behind formal abstraction. So while doctrines formally separate liability along those lines, where the boundary is porous, liability flows past the boundary as well. It should also not be surprising that in asking courts to examine technology, we end up mirroring doctrines from areas of law like products liability. Those areas already consider technological design, and we're asking courts to apply the same principles more broadly.

In sum, the responsibility for the discriminatory outcome of data-driven hiring could properly lie with the employer, the software company, both, or neither. Where the technology company is not part of the lawsuit and the employer lacks technical expertise, it will often be tempting to treat the technology as a fixed object, and to hold that the only choices available to the employer were to use the technology or not. Courts should resist automatically treating the technology that way and interrogate the alternatives that could have existed. Maybe the defendant employer really is not at fault—maybe they shopped around, or the technology was advertised as less discriminatory than it turned out to be—in those cases, the defendant would rightfully be off the hook, but the decision would have been made by the court, not the technologists who built the product.

C. Criminal Law

Technology is increasingly being used to support decisions in the context of criminal justice—bail, sentencing, and parole.[195] If government officials are to consider the use of predictive tools to inform decision-making in the criminal justice system, they must evaluate whether the values and assumptions embedded in the design of those tools are appropriate for the given context. And if courts are to use those tools, they must also be aware of

[194] Restatement (Third) of Torts: Prod. Liab. § 5 (1998).

[195] Hannah S. Laqueur & Ryan W. Copus, *Machines Finding Injustice*, 23 SMU Sci. & Tech. L. Rev. 151, 155 (2020) ("Jurisdictions across the country are increasingly deploying risk prediction algorithms to aid in decision tasks such as setting bail, determining sentencing, and deciding whether or not to release on inmates on parole.").

those design choices so they can appropriately apply their discretion with the information the tools provide.

The decision in *State v. Loomis*[196] illustrates how critical this analysis is. In this case, the plaintiff argued that the use of a proprietary risk assessment algorithm in his presentencing report was a violation of due process, but the Wisconsin Supreme Court disagreed.[197] An important component of the Court's reasoning was the fact that the report was not the sole basis of the judge's decision, because "courts have the discretion and information necessary to disagree with the assessment when appropriate."[198] But for this information to be as complete as possible, it must include an analysis of the specific implementation of the assessment technology and the implications for the outputs. How can a court know whether to trust or challenge the output of these assessments without at least trying to understand the implications of the technical details? While it may not be especially practical for a judge to conduct these analyses during individual pretrial or sentencing decisions—which are usually made quickly[199]—these considerations are also important when making the administrative decision of whether to deploy the tool at all.

A recent paper by Pascal D. König & Tobias D. Krafft examines the question of what judges and policy-makers need to know to effectively incorporate evidence from risk assessment algorithms.[200] Using an extensive survey an of actual algorithmic decision-making tools in the criminal justice system, they explain the many "dimensions" from which the tools should be assessed and compared, most of which boil down to which design decisions were made and their effects; we discuss some of them here.

Consider a common application used to inform bail recommendations: a model trained to predict a defendant's failure to appear (FTA) in court. One can imagine two versions of such a tool: a forecasting tool meant only to predict the FTA outcome, or a tool that attempts to isolate more causal mechanisms, quantifying the effects of potential hidden variables

[196] State v. Loomis, 881 N.W. 2d 749 (Wis. 2016)

[197] *Id.*

[198] Case Note, *State v. Loomis*, 130 HARV. L. REV. 1530 (2017).

[199] Megan Stevenson & Sandra G. Mayson, *Bail Reform: New Directions for Pretrial Detention and Release*, *in* ACADEMY FOR JUSTICE, A REPORT ON SCHOLARSHIP AND CRIMINAL JUSTICE REFORM (Erik Luna ed., 2017); Cynthia E. Jones, *"Give us Free": Addressing Racial Disparities in Bail Determinations*,16 LEGISLATION AND PUB. POL'Y, 919 (2014).

[200] Pascal D. König & Tobias D. Krafft, *Evaluating the evidence in algorithmic evidence-based decision-making: the case of US pretrial risk assessment tools*, 33 CURRENT ISSUES IN CRIMINAL JUSTICE 359, 359-381 (2022).

that could affect the outcome and suggesting possible interventions to change it. Suppose the tool was the former—built with traditional machine learning and pattern recognition techniques which are optimized to compute accurate predictions based on historical data. This type of tool can only provide an estimate of whether the defendant will appear if no actions are taken on behalf of the court to help the defendant appear. This is true even if the model itself has inputs that can be intervened upon, as is often the case in real tools.[201] For instance, perhaps our model is linear, and the learned coefficient for an individual's employment status is high. Perhaps the model categorizes some unemployed defendant with an otherwise clean record as "high-risk," and the breakdown of the model coefficients shows the output can largely be attributed to this factor. This would still reflect an associative, rather than necessarily causal, relationship in the data: the model does not imply that assisting the defendant with their employment status would be an effective way to help them appear for their court date.

Despite the prevalence of this technique in actual criminal justice settings[202], statistical methods like "propensity score matching" exist to build models that estimate the causal effects of interventions.[203] Such techniques are being explored more actively in other contexts. In child welfare, for example, the problem of observation bias—the fact that only families that get child protective services called on them end up with evaluations—have inspired those who work on risk assessments for welfare cases to introduce modeling frameworks which take these biases into account. Amanda Coston and colleagues have worked on models which focus on quantifying the causal effect of visiting a family with a potential abuse case.[204] Turning back to criminal justice, then, if a court is presented with a forecasting tool, the government must decide whether this framework is aligned with its policy goals. Forecasting tools are not designed to identify ways to improve outcomes, and if improving outcomes is their goal, they should reject it. Additionally, if a judge presented with information from a forecasting tool understands this difference, they will be better equipped to evaluate whether that information should be used to inform their choice of bail conditions. But

[201] Chelsea Barabas et al, *Interventions Over Predictions: Reframing the Ethical Debate for Actuarial Risk Assessment*, 81 PROCS MACH. LEARNING RSCH. 62, 72-74 (2019).

[202] König & Krafft, *supra* note 200.

[203] *See* Austin Nichols, *Causal Inference with Observational Data*, 7 THE STATA J. 507, 516 (2007); Barabas, et al, *supra* note 201, at 70.

[204] *See, e.g.*, Amanda Coston et al., *Counterfactual Risk Assessments, Evaluation, and Fairness*, PROC. 2020 ACM CONF. ON FAIRNESS, ACCOUNTABILITY, AND TRANSPARENCY (2020).

if courts take the technology as a given, unaware that other options exist, they may accept a tool that doesn't act according to policy as preferable to no tool at all.

Another design choice that must be examined in the context of this particular model is the target variable in the data used for trainings: FTA at a single court appearance. A defendant can miss court for any number of reasons, which may or may not be intentional on their part. In fact, it has been shown that simple design changes in criminal summons and text message reminders increase the likelihood that defendants show up to their appointed court date, suggesting that simple human error causes a number of FTAs.[205] However, the risks that are actually considered important by courts in pretrial liberty are flight from prosecution and *willful* failure to appear,[206] an outcome that is not perfectly captured by this target variable. A model that appears to be accurate for overall FTAs could be poorly predictive of willful FTAs.[207] A better indication for the risks that the court actually cares about may or may not have been present in the data the tool designers had access to. Even if this data was available, designers may have deliberately conflated willful or non-willful FTAs for other modeling reasons, such as wanting to avoid learning problems induced by imbalanced data (if, for instance, willful and non-willful FTAs had very different frequencies). With enough information, a judge or government agency could conclude that the particular choices made by the designer were misaligned to the criminal justice system's goals.

The case of the Colorado Pretrial Assessment Tool presents one such example. After the tool was created, the University of Northern Colorado conducted a validation study, and after finding bias issues, proposed an amended tool, the CPAT-R.[208] Both tools, however, only attempt to predict whether a pretrial defendant will miss a single court appearance or be charged with an arrestable offense. The ACLU of Colorado subsequently analyzed

[205] Alissa Fishbane, Aurelie Ouss, & Anuj K. Shah, *Behavioral Nudges Reduce Failure to Appear for Court*, 370 SCI. 682 (2020).

[206] *Dangerous, Misleading and Biased: A Letter on Pretrial Risk Assessment Tools in Colorado*, ACLU of Colorado (Oct. 29, 2020), https://www.aclu-co.org/sites/default/files/Dangerous-Misleading-and-Biased-A-Letter-on-Pretrial-Risk-Assessment-Tools-in-Colorado.pdf.

[207] König and Krafft make a similar point in the context of risk assessments which predict composite variables that combine the outcomes of "failure to appear" and "new crime/arrest." König & Krafft, *supra* note 200.

[208] VICTORIA A. TERRANOVA & KYLE C. WARD, COLORADO PRETRIAL ASSESSMENT TOOL VALIDATION STUDY (2020)

the design of both the original and proposed tools.[209] They pointed out that the validation study's authors had demonstrated the ability to differentiate between FTAs based on whether a court sanction was imposed after the fact and argued that these "high-consequence" FTAs were common enough to use as a target variable.[210] That they chose not to target the "high-consequence" FTAs in their model was a design choice that the ACLU argued rendered the tool less aligned with the purpose of pretrial risk assessment.[211] On top of this, the CPAT-R considered an individual's prior FTAs as an additional input, resulting in the substantial overprediction of risk for Black and Hispanic people compared to white defendants.[212] For these reasons, among other validity issues, the ACLU argued that neither the CPAT nor CPAT-R should be used at all, but without unpacking the tool, courts setting bail might not know that these problems—and other options—exist.

Another choice made during the development of a risk assessment tool is how to convert model outputs into recommendations, a step sometimes referred to as post-processing. Different post-processing methods can have different impacts on measures of both fairness and accuracy of a model, and therefore the choice of method encodes policy decisions that should be made by the court. Yet this step is often overlooked or unacknowledged.[213]

When a model produces a quantitative risk score, a single decision recommendation can only be made if the developer of the tool chooses a threshold above which a defendant is considered a likely enough FTA risk to warrant a particular bail decision. This choice directly affects performance measures which are based on how many individuals the model misclassifies (*e.g.*, true positive rate, false positive rate, accuracy). A threshold may be chosen to minimize or maximize a subset of these metrics. As König and

[209] ACLU of Colorado, *supra* note 206, at 8.

[210] *Id.*

[211] *Id.*

[212] *Id.*

[213] In prior work, two of us (Selbst & Venkatasubramanian) referred to this exact error as the "framing trap" in a discussion of abstraction errors that computer scientists fall into when designing fair machine learning systems. *See* Selbst, et al., *supra* note 36, at 60. There, we argued that data scientists get the abstraction wrong if they fail to account for judicial behaviors when making fairness guarantees, because fairness is a property for the entire system—the technical and the court—and by making guarantees about only the technical system, their guarantees will not hold. *Id.* Here, our argument comes full circle, as judges who defer to a technical system with incorrect fairness guarantees end up failing in their role *because* the system designers made choices that should be interrogated.

Krafft put it, in the context of a tool which estimates the predicted probability of some event coded as a binary indicator (0 or 1):

> If one were to set the threshold to 0.7, for instance, this would set the bar higher for the positive class (1): making sure that a case classified as 1 is correctly predicted as such (avoiding false positives). Conversely, a lower threshold means being more acceptant of classifying cases as 1 even though they are a 0 (avoiding false negatives and missing fewer actual positives). In sum, the threshold directly affects which classification error is taken as more acceptable. In substantive terms, this boils down to the question: is it more important to get one's positive predictions right or to find all actual positive cases?[214]

This makes a choice of threshold (and the corresponding error metrics) not only a normative policy decision, but an indicator of how to interpret the recommendation. From the days of Blackstone, the proper ratio of false negatives to false positives in criminal justice has always been the prerogative of the legal system,[215] and it should not now default to whatever a developer happened to choose when creating the technology.

CONCLUSION

The metaphor of a black box is frequently used to describe technology, especially software that relies heavily on machine learning or artificial intelligence. When these technologies have their day in court, decisions are typically evaluated by examining the facts on the ground and lines of responsibility defined by the technology as presented. In effect, the technology is viewed as a design *fait accompli*.

In this Article, we have argued that this is a mistake. Black boxes are constructed via a series of design choices that serve a dual purpose. First, the design choices have the effect of moving elements of the design inside the technology. Second, the choices shift certain elements of the design—those chosen to be outside the black box's operating interface—to the outside environment. These decisions construct a concept that becomes understood

---

[214] König & Krafft, *supra* note 200, at 372.

[215] Daniel Epps, *The Consequences of Error in Criminal Justice*, 128 HARV. L. REV. 1065, 1067 (2015) (Discussing the continued importance of the famous "Blackstone Principle" that it is "better that ten guilty persons escape, than that one innocent suffer." (quoting 4 WILLIAM BLACKSTONE, COMMENTARIES *352)).

as the technology, establishing the facts about how it works and who is responsibility for its workings and interactions with the world.

Courts are accustomed to interrogating technological design choices in settings that clearly present legal questions about technological design, such as in product liability cases and intellectual property regimes. But as technical designs increasingly underpin critical decision-making settings such as in criminal justice, employment, credit, housing, medicine, and many other contexts, it is imperative that courts start to deconstruct design choices in ML and AI-oriented technologies as well. This deconstruction has a material effect on adjudications of all types. While design choices are driven by considerations such as efficiency, scalability, and cost, they nonetheless have a profound impact on how cases involving that technology might be resolved, and which entities are held liable, but such an impact is avoidable if courts pay attention to them.

Of course, none of this is easy for courts, and much of the necessary information may be hard to find. Going forward, technologists need to be far more transparent about the nature of the choices they make when designing new technology. They must create detailed documentation of the design choices made and the rationales for them.[216] In turn, courts need to be prepared to ask more specific questions about technology design choices and require explanations of how these choices might have led to the outcome being adjudicated, and what alternative design strategies might have been suitable. This will have feedback effects as well. If courts demonstrate a willingness to deconstruct design, designers will have a greater incentive to design transparently to reduce litigation costs and will ultimately design more thoughtfully to incorporate not just efficiency and scaling considerations, but potential harms that they might have to litigate later.

Courts cannot adequately perform their judicial role if they consistently view technology as a black box. A rigorous interrogation of design decisions is necessary for courts to accurately assign liability and vindicate the

[216] *See, e.g.*, Andrew D. Selbst, *An Institutional View of Algorithmic Impact Assessments*, 35 HARV. J. L. & TECH. 117 (2021) (discussing the benefits and challenges of algorithmic impact assessments); Emanuel Moss, Elizabeth Anne Watkins, Ranjit Singh, Madeleine Clare Elish & Jacob Metcalf, Assembling Accountability: Algorithmic Impact Assessment For The Public Interest (2021); Margot E. Kaminski & Gianclaudio Malgieri, *Algorithmic Impact Assessments Under the GDPR: Producing Multi-layered Explanations*, 11 INT'L DATA PRIVACY L. 125, 125 (2020); Sonia K. Katyal, *Private Accountability in the Age of Artificial Intelligence*, 66 UCLA L. Rev. 54, 99 (2019).

normative goals of the laws they interpret and apply. If courts can learn to deconstruct design, we will all be better off.