COMMUNICATIONS OF THE ACM

CACM.ACM.ORG

02/2024 VOL.67 NO.02

Inherent Limitations of AI Fairness

Service Robot Anthropomorphism

Gaining Benefits from AI and Data Science

Computing Education in the Era of Generative AI

Talking about Large Language Models

Association for Computing Machinery

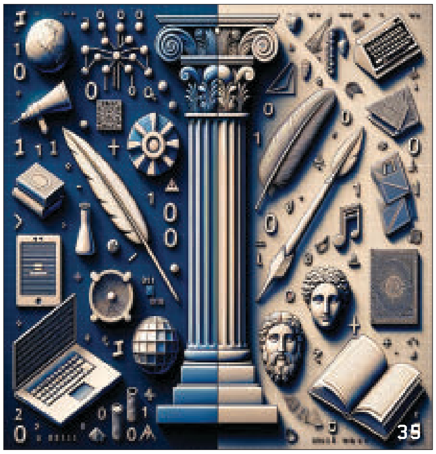
COMMUNICATIONS OF THE ACM

News

Last Byte

Opinion

Opinion



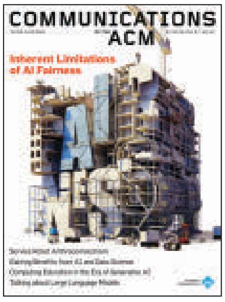
Watch the author discuss this work in the exclusive *Communications* video. https://cacm.acm.org/videos/gaining-benefit

LEFT AND RIGHT IMAGES CREATED BY ANDRIJ BORYS ASSOCIATES WITH THE ASSISTANCE OF SHUTTERSTOCK.AI

Association for Computing Machinery
Advancing Computing as a Science & Profession

Practice

Articles' development led by acmqueue
queue.acm.org



About the Cover:
Ensuring AI is built upon a foundation of fairness is critical to the many highly sensitive applications in which it is being called upon to make high-stakes decisions. In this article, the authors delineate the role of fair AI as well as the mechanisms that can help it fulfill its promises in realistic settings. Cover by Peter Crowther Associates.

Research

Watch the authors discuss this article in the exclusive *Communications* video.
https://cacm.acm.org/videos/inherent-limitations

Research

Research Highlights

LEFT IMAGE BY MARYNOVA; RIGHT IMAGE CREATED BY ANDRIJ BORYS ASSOCIATES WITH THE ASSISTANCE OF SHUTTERSTOCK.AI

COMMUNICATIONS OF THE ACM

Trusted insights for computing's leading professionals.

Communications of the ACM is the leading monthly print and online magazine for the computing and information technology fields. *Communications* is recognized as the most trusted and knowledgeable source of industry information for today's computing professional. *Communications* brings its readership in-depth coverage of emerging areas of computer science, new trends in information technology, and practical applications. Industry leaders use *Communications* as a platform to present and debate various technology implications, public policies, engineering challenges, and market trends. The prestige and unmatched reputation that *Communications of the ACM* enjoys today is built upon a 50-year commitment to high-quality editorial content and a steadfast dedication to advancing the arts, sciences, and applications of information technology.

ACM, the world's largest educational and scientific computing society, delivers resources that advance computing as a science and profession. ACM provides the computing field's premier Digital Library and serves its members and the computing profession with leading-edge publications, conferences, and career resources.

Association for Computing Machinery (ACM)
1601 Broadway, 10th Floor
New York, NY 10019-7434 USA
T (212) 869-7440; F (212) 869-0481

Subscriptions
An annual subscription cost is included in ACM member dues of $99 ($40 of which is allocated to a subscription to *Communications*); for students, cost is included in $42 dues ($20 of which is allocated to a *Communications* subscription). A nonmember annual subscription is $269.

Single Copies
Single copies of *Communications of the ACM* are available for purchase. Please contact acmhelp@acm.org.

COMMUNICATIONS OF THE ACM
(ISSN 0001-0782) is published monthly by ACM Media, 1601 Broadway, 10th Floor New York, NY 10019-7434 USA. Periodicals postage paid at New York, NY 10001, and other mailing offices.

POSTMASTER
Please send address changes to *Communications of the ACM*
1601 Broadway, 10th Floor
New York, NY 10019-7434 USA

Printed in the USA.

Association for Computing Machinery

PLEASE RECYCLE THIS MAGAZINE

DOI:10.1145/3638528

Vinton G. Cerf

Validating Factual Personal Information

We are called upon, frequently, to validate personal information: name, address, phone number, birthdate, national identification number, birthplace, employment, educational record, and more. Often this demand is made online. Browsers can capture some of this information and automatically fill it in. It is not a new idea to try to automate this. Moreover, as facts change (for example, a new address, new phone number, new employment), we might find it useful to automatically propagate this information to places where earlier versions have been registered. How is the registry to know whether the information is accurate? How can there be control over the release of this information? I have been wondering what properties would be useful to realize in a system intended to keep personal information, where it is needed, up to date. What follows is only partly digested and reader reactions would be appreciated.

First, let us suppose all this information can be structured as name:value pairs and that the names are widely standardized as to their meaning. One could imagine a business which records this data and validates it according to accepted (and maybe legislated) practices and releases it only on authorization by the party submitting it in the first place. Since this business is to be trusted to validate and disgorge information only on authorization, one might imagine that such a company would have to be certified somehow and pass rigorous tests of its ability to control access to and the release of such personal information. One would imagine the possibility of widely accepted safety and security standards analogous to generally accepted accounting practices (GAAP).

One could imagine that the initial cache of personal data might have to be submitted in person as is sometimes required when establishing bank and securities accounts, driver's licenses, passports, or corporate identification badges. In any case, there would have to be criteria for vetting the information so that the attestation of its accuracy is accepted as valid by relying parties.

The registrant would bring verifiable data to the registrar, who would validate and record the information. At this time, the registrant would generate or be given a public/private key pair. The public key would be shared with the registry. A relying party needing valid personal information (for example, age, birthdate, birthplace, current residence, email address, and phone number) would request it from the registry. The registry issues a request to the registrant to authorize release of the data, which includes identifying the party making the request. The registrant should receive sufficient information to validate the relying party and its intended use of the data. The registrant can then authorize—for example, by digital signature—the release of the data by the registry to the relying party.

What properties would be useful to realize in a system intended to keep personal information, where it is needed, up to date?

Another use of such a system is for the registrant to send updated information to the registry. Relying parties might, by practice, automatically query the registry whenever personal information is needed so that updates propagate when a request is made. The relying parties might make periodic queries to keep their information up to date or they might keep no substantive information but, rather, request it when needed. It is obvious that the data transfers should be encrypted for privacy and digitally signed by the registry to assure data integrity. Registrants could individually permit automatic responses to queries by specific relying parties, or they might insist on authorizing the release of access every time a request is made.

Updates to the registered data could be validated both by digital signature of the registrant as well as through vetting of the new data by the validating registry. There might be different levels of validation available, not unlike real-estate title searches and various certificate authorization practices. The business model for such a system might include subscription fee payments by the registrants and transaction fee payments by parties requesting valid personal data.

It will not surprise me to find that such services already exist. I will be interested to learn from readers what they think about this idea and perhaps what risk factors should be considered.

Vinton G. Cerf is vice president and Chief Internet Evangelist at Google. He served as ACM president from 2012–2014.

BLOG@CACM

The *Communications* website, https://cacm.acm.org, features more than a dozen bloggers in the BLOG@CACM community. In each issue of *Communications*, we'll publish selected posts or excerpts.

twitter

Follow us on Twitter at http://twitter.com/blogCACM

DOI:10.1145/3638537 https://cacm.acm.org/blogs/blog-cacm

Why Bother Localizing Information Technology Products?

Alex Tray offers advice and insight into how and why to localize information technology products.

Alex Tray
How to Ace IT Product Localization
https://bit.ly/3Qoxibq
October 19, 2023

As information technology (IT) keeps bridging geographic gaps and connecting people from different cultures, the demand for IT product localization has never been more pressing.

Localization not only improves the user experience, but also makes a significant contribution to an organization's global success by making IT products more accessible and user-friendly to multinational audiences.

This post explores what IT product localization is, including how it works and the several benefits it offers businesses that want to expand their presence in foreign markets.

What Is IT Product Localization?

IT product localization involves tailoring software, websites, and digital content to suit the specific needs and preferences of users in various regions and cultures.

It goes beyond mere translation. It also entails a thorough adaptation that considers language, cultural, and technical factors to ensure the product is not only usable but also appealing to customers in the target market.

Key Benefits of IT Product Localization

Localizing IT products has several essential benefits for businesses that want to grow internationally and serve diverse markets. These benefits include:

Improved market accessibility. Localization allows your IT products to expand into new areas and connect with customers who might not be native speakers of the original product's language. Language barriers are eliminated when your software, apps, or tech solutions are made available in the language of your target audience.

This increases sales and brand recognition by not only expanding your market reach but also making your products more accessible to potential buyers.

Enhanced user experience. Customizing the user experience to fit the interests and cultural norms of your target audience is a crucial component of localizing IT products. Adapting the interface, content, and user interactions can result in a more intuitive and engaging user experience.

This adaptation indicates your dedication to knowing about and helping your target audience, ultimately enhancing consumer loyalty and confidence.

Increased global market share. Effective IT product localization may have a significant impact on your global market share. You may outperform regional businesses and well-established global competitors by adapting your IT products to local markets.

Customers like your locally adapted products, which gives you a competitive advantage, expands your market share, and eventually boosts your sales.

Customer satisfaction and loyalty. Satisfied customers are more inclined to stick with a business. Localizing IT products shows you care about the needs and preferences of your clients. This builds customer loyalty and promotes a favorable perception of your business.

Customers who are happy with your products are more likely to not only keep using them but also promote them to others, thereby boosting organic growth.

Factors to Consider When Localizing IT Products

There are several factors that you need to consider before localizing IT products. These include:

Cultural nuances. Every region has its traditions, customs, and even sense of humor. To ensure your product appeals to the local market, it should respect and align with these factors.

This may entail adapting the graphics, colors, symbols, and even the user interface to make the user interface more culturally appropriate.

Quality assurance. Technical issues, such as broken code, unsuitable typefaces, or problems with user interface elements, usually occur during the localization process. Thorough quality assurance testing is required to find and fix these problems before the locally adapted product hits the market.

Regular testing and debugging are essential to ensure the IT product operates accurately in the target region and language.

Adaptation to local preferences. Beyond linguistic and cultural factors, it is important to consider regional preferences and user expectations. Regional differences can be significant in how people use technology. For instance, there may be differences in navigational patterns, payment options, and design aesthetics.

User acceptance and satisfaction can be significantly increased by being aware of and embracing these preferences.

Internationalization and globalization. Internationalization involves building your IT product in a way that makes localization easy. This involves using flexible coding techniques and structuring content to be language-neutral for easy market adaptation.

The goal of globalization, on the other hand, is to develop products that are scalable and flexible for a global audience. The time and effort needed for localization can be greatly reduced with a solid internationalization and globalization strategy.

Localization tools and technologies. The use of the right localization tool and technology can improve the accuracy and efficiency of the localization process. The localization workflow can be made more efficient with the aid of translation memory systems, content management systems, and localization management platforms such as Centus.

Every region has its traditions, customs, and even sense of humor. Your product should respect and align with these factors.

These solutions make it easier to manage translations, ensure consistency throughout the entire product, and effectively track changes. A successful localization strategy relies on choosing the right tools and technological stack.

How to Implement an IT Product Localization Strategy

You can ensure your IT product is successful in international markets by considering these steps:

Conduct market research. Conducting thorough market research is essential before starting the localization process. This requires locating your target markets and getting to know their specific preferences, cultures, and languages.

You can efficiently adapt your product to meet local needs by conducting market research to gain essential information. You can use it to assess possible demand and competition in the markets you have chosen.

Build your IT product localization team. The foundation of successful IT product adaptation is the creation of a talented and diversified localization team. Native speakers of the target languages, cultural experts, software developers, and quality assurance professionals should ideally be on your team.

Apart from maintaining the technical integrity of your IT product, a well-rounded team can successfully manage the technicalities of linguistic and cultural differences.

Use relevant software and expertise. Use modern localization techniques and technologies to speed up the process. Collaboration platforms, content management systems, and translation memory systems can all improve output and consistency.

If you do not have an in-house expert, collaborate with localization specialists or agencies. Their expertise can help you save time and steer clear of typical errors.

Localize for each target market. Remember that when localizing IT products, a one-size-fits-all strategy rarely succeeds. Adapt your products for each market, considering not only linguistic requirements, but also cultural preferences and regional laws.

This can involve translating the user interface, supporting documents, and marketing collateral for your product, as well as adapting features and functionality to meet regional demands.

Test the IT product localization results. An important phase in localizing IT products is quality assurance. To ensure your product operates flawlessly in each target market, thorough testing is important. This covers usability assessments as well as language and functional testing.

You can use feedback from local users during beta testing to find and fix any problems that may have gone unnoticed during localization.

Conclusion

Customers around the world want a flawless user experience in their own language. Therefore, businesses that localize their IT products find it easier to win over new target markets.

Given the importance of localization, your focus should be on how to localize your IT product, not whether you should.

To streamline your IT product localization process, conduct thorough market research, attract local talent, and use professional localization management platforms.

Alex Tray is a system administrator and cybersecurity consultant with 10 years of experience. He is currently self-employed as a cybersecurity consultant and as a freelance writer.

N news

Science | DOI:10.1145/3637207

Erica Klarreich

Algorithmic Advance: The Group Isomorphism Problem

Exploring a potential way to immensely speed up algorithms for the group isomorphism problem.

A PAPER POSTED ONLINE in March 2023 has presented the first substantial progress in a half-century on one of the fundamental questions in the overlap between mathematics and computer science: how to recognize when two "groups"—basic algebraic structures—are really the same group in disguise.

Groups, which have been studied by mathematicians since the late 18th century, are among the most ubiquitous structures in mathematics. A group is a set of elements together with an operation (such as addition, multiplication, or composition) that turns two elements into a new one. To qualify as a group, the set must contain an "identity" element (one that leaves every element unchanged when combined with it by the operation); every element must have an inverse; and the operation (usually written *) must be associative, meaning that (a*b)*c always equals a*(b*c). There are, for example, groups of numbers or matrices or permutations, groups of symmetries of some object, and groups that measure the topological features of a shape.

A full classification of groups—even just the finite ones—is probably forever out of reach, said Bettina Eick, a mathematician at the Technical University of Braunschweig in Germany. A more attainable goal is to determine, for a given pair of finite groups, whether they are "isomorphic"—that is, whether there is a way to match up their elements that respects the behavior of their respective group operations. This "group isomorphism" problem was formulated more than a century ago by the mathematician Max Dehn, who was interested in, among other things, using groups to distinguish between different knots. Group isomorphism is "a very central problem, that's for sure," Eick said.

It is always possible, in theory, to

IMAGE BY BAIVECTOR

test whether two finite groups are isomorphic, but doing so may take an infeasible amount of time when the groups are large. The group isomorphism problem is not known to be in P, the set of problems with algorithms that run in a polynomial amount of time—what theoretical computer scientists think of as the "easy" problems.

From a computer science perspective, the runtime of group isomorphism algorithms connects with a host of other questions. The problem has been proposed as the basis for several cryptosystems that might withstand attacks from a quantum computer. And the intractability of the group isomorphism problem is a barrier to making progress on the closely related graph isomorphism problem, which strives to identify when two graphs (that is, networks) are the same.

Given two finite groups, it is simple to check whether a proposed isomorphism between them really is one. This means the group isomorphism problem belongs to the complexity class known as NP, consisting of problems whose solutions are easily checked. But group isomorphism occupies an unusual position within this class. It is neither known to be in P nor known to be NP-complete—the hardest kind of problem in NP. Instead, group isomorphism is one of just a few natural problems that, as far as we know, seem to sit somewhere in the gulf between these two extremes. "The problem is not just natural, but also pretty unique," said Xiaorui Sun, a computer scientist at the University of Illinois Chicago.

Now, Sun has come up with a significantly faster algorithm for group isomorphism than the previous best one (though still not fast enough to put the problem in P). At present, his algorithm works only for a certain key subclass of groups, but computer scientists are optimistic they will be able to extend his insights to broader classes of groups.

"The ice has been broken," said Laszlo Babai, a computer scientist at the University of Chicago. "After 50 years, we finally have something substantial to talk about."

A Bottleneck

Given two groups, G and H, an isomorphism between them is simply a pairing of their elements (call it f) that respects the algebraic structure of the groups, so that if $a * b = c$ in G, then $f(a) * f(b) = f(c)$ in H. In other words, f transforms the "multiplication" table in G into the multiplication table in H.

One way to determine whether two finite groups are isomorphic is simply to check every possible pairing of their elements. This brute force approach takes an exponential amount of time, making it impractical for all but the smallest groups.

In the 1970s, however, computer scientist Robert Tarjan noted you can speed things up by first constructing a list of "generators" for G—elements which, when multiplied together in different combinations, produce every other element of G. Then, instead of considering every mapping from G to H, you can consider every mapping of these generators into H, and see if any such mapping extends to an isomorphism. There's always a generator list with at most about $log(n)$ elements (where n is the number of elements of G), giving this algorithm a runtime of roughly $n^{log(n)}$—drastically speedier than the exponential algorithm, though modestly slower than a polynomial runtime.

Since that observation by Tarjan, computer scientists have been more or less stuck. They have managed to make some small improvements to the $n^{log(n)}$ runtime, but essentially, "there's been no progress for about half a century," Sun said, even though "almost everyone in the field believes that there truly exists a faster algorithm."

There is not much room left for computer scientists to improve graph isomorphism unless they can speed up group isomorphism.

Any time you are trying to compare two finite groups, there is a way to translate the problem into an analogous problem about comparing two graphs—in other words, the group isomorphism problem reduces to the graph isomorphism problem. That means graph isomorphism is at least as hard to solve as group isomorphism. And indeed, for most of the history of the two problems, the best known algorithm for graph isomorphism was much slower than the best known algorithm for group isomorphism.

But in 2015, Babai managed to construct an algorithm for graph isomorphism whose runtime is nearly as fast as the $n^{log(n)}$ runtime for group isomorphism. At this point, there is not much room left for computer scientists to improve graph isomorphism unless they can speed up group isomorphism. "Our inability to improve group isomorphism testing remains a bottleneck for graph isomorphism," Babai said.

The Hardest Cases

Sun's paper has offered computer scientists a potential way through this bottleneck. He has come up with a group isomorphism algorithm whose runtime is roughly $n^{(log(n))\hat{}5/6}$, a significant speedup over $n^{log(n)}$. His approach applies not to all finite groups, but to a core collection of groups known as "p-groups of class 2 and exponent p." Loosely speaking, these requirements mean (among other things) that the number of elements in the group is a prime number p raised to some power, and the group operation almost (but not quite) follows the commutative law, which says that $a * b = b * a$.

These might seem like esoteric restrictions, but p-groups of class 2 and exponent p play a central role in understanding the structure of finite groups as a whole. For one thing, these groups sit at the bottom of a natural hierarchy of all finite groups, offering the hope that methods developed for this case might be extendable to higher levels of the hierarchy.

Also, p-groups of class 2 and exponent p have historically stymied attempts at a fast group isomorphism algorithm. For groups where the operation is precisely commutative, the isomorphism problem is easy. And for groups that are far from commutative, it's often possible to latch onto their complex structure to solve the isomorphism problem. But p-groups of class 2 and exponent p sit in an awkward trough between these two possibilities, lacking either the simplicity or the complexity that makes those other groups tractable.

"Many people believe these are the hardest cases of group isomorphism," said Joshua Grochow, a computer scientist at the University of Colorado Boulder.

To deal with these tricky groups, Sun starts by invoking a correspondence discovered in the 1930s between groups of this type and certain spaces of matrices. That means constructing an isomorphism between two such groups boils down to constructing a similar correspondence between their associated matrix spaces. Previously, the fastest known method for comparing these matrix spaces was just slightly faster than examining all the possible maps between them; Sun figured out how to do better.

His method begins by using a preexisting technique called "individualization and refinement" to create for each matrix a smaller matrix that serves as a sort of label for it. Others had tried this type of technique before, but they all ran into the same stumbling block: if you make the "labels" small enough to be able to compare them efficiently, then there aren't enough of them to give each large matrix a different label. To deal with this difficulty, Sun had to figure out how to handle "low rank" matrices in the matrix spaces—matrices that, thought of as linear transformations, compress high-dimensional spaces into much lower-dimensional spaces. One of his key insights was how to make the low-rank portion of these matrix spaces more organized.

"This method feels really new," Grochow said.

Sun's result is a purely theoretical advance—it gives speed guarantees as the size of the group approaches infinity but may not beat out existing algorithms in the size regimes that arise for mathematicians using software packages to compare groups. Nevertheless, algorithms with new structural insights, such as Sun's, often seem to perform better in practice than their guarantees give them a right to, Babai said. If Sun's algorithm does work well in practical settings, Eick said, it should be possible to embed it in other practical algorithms to cover a much broader category of groups than the one Sun studied.

LASZLO BABAI, UNIVERSITY OF CHICAGO

"The new energy Sun's work has unleashed is … palpable."

Computer scientists are hastening to extend Sun's theoretical result. Already, Grochow and Youming Qiao, of the University of Technology Sydney, have figured out how to loosen the "class 2" restriction, and researchers hope to go farther, gradually extending Sun's result up much of the hierarchy of groups.

"The new energy Sun's work has unleashed is … palpable," Babai said.

Further Reading

Babai, L.
Graph isomorphism in quasipolynomial time. ***ACM Symp. on Theory of Computing (2016)***, **684–697.**

Grochow, J.A. and Qiao, Y.
On the complexity of isomorphism problems for tensors, groups, and polynomials IV: linear-length reductions and their applications. https://arxiv.org/abs/2306.16317

Ivanyos, G. and Qiao, Y.
Algorithms based on*-algebras, and their applications to isomorphism of polynomials with one secret, group isomorphism, and polynomial identity testing. ***SIAM J. on Computing 48***, **3 (2019), 926–963.**

Sun, X.
Faster isomorphism for p-groups of class 2 and exponent p. https://arxiv.org/abs/2303.15412

https://en.wikipedia.org/wiki/Group_isomorphism

Erica Klarreich is a mathematics and science journalist based in Berkeley, CA, USA.

ACM Member News

ADVANCING HUMAN-COMPUTER INTERACTION

Michel Beaudouin-Lafon is professor of Computer Science, Classe Exceptionnelle, at Université Paris-Saclay in Paris, France. Beaudouin-Lafon received the equivalent of his master's degree in Computer Science and Applied Mathematics from ENSEEIHT, France. He earned his Ph.D. in Computer Science from Université Paris-Sud (now Université Paris-Saclay) in 1985, then joined the school's faculty, where he has remained.

Human-computer interaction has been the primary focus of Beaudouin-Lafon's research throughout his career, including fundamental aspects of interaction, novel interaction techniques, engineering interactive systems, and collaborative computing.

"At the moment, I am co-directing a French research project called eNSEMBLE," a network of more than 100 research groups whose aim is to make computing truly collaborative, instead of the current trend of simply adding collaborative features to existing systems. "eNSEMBLE includes social scientists," Beaudouin-Lafon says, "because you can't develop technology if you don't understand the people who are going to use it."

Beaudouin-Lafon has been involved with ACM in various capacities over the years. He was a founding member of the ACM Europe Council and is currently vice chair of the ACM Technology Policy Council, responsible for the organizations' global policy initiatives. "I always found myself interested in policy, educating and communicating to people and policymakers the possibilities and risks of technology," he says. He believes this is especially important now, with generative AI and the environmental impact of IT becoming increasingly critical issues.

—John Delaney

Technology | DOI:10.1145/3637208

Chris Edwards

Teaching Transformed

The apparent ability of LLMs to write functioning source code has caused celebration over the potential for massive increases in programmer productivity and consternation among teachers.

As owner of GitHub and lead investor in OpenAI, the developer of the GPT-x series of large language models (LLMs), it did not take long for Microsoft to see the potential for collaboration between the two. Three years ago, GitHub partnered with OpenAI to develop Codex as an automated assistant for programmers, quickly followed by the Copilot code-completion tool. The public release of ChatGPT by OpenAI toward the end of 2022 made the technology even more widely available to software developers and people learning to program, with other vendors joining in the effort to automate the job of writing software using LLMs.

Rapid scaling has enabled major improvements in the ability of artificial intelligence (AI) to turn natural-language requests into working code. Workplace studies have claimed LLMs boost productivity on real-world projects. In its own survey of usage by close to a million users over the year since the launch of Copilot, GitHub claimed developers accepted on average 30% of the tool's code suggestions and that usage of the suggestions increases over time as programmers become more familiar with the tool's recommendations.

For its own tests, management consultancy McKinsey recruited around 40 developers working in-house across the U.S. on two types of LLMs fine-tuned on coding problems. The experiment showed the tools could halve the time to write new code, with an average speed-up of around a third.

Though employers might welcome the productivity improvements LLMs promise, educators are concerned about how they might impact their ability to test students' understanding. Carnegie Mellon University postdoctoral fellow Jaromír Šavelka and colleagues analyzed the difference between GPT-3, which formed the basis for ChatGPT, and GPT-4, released spring 2023, on their ability to answer questions from three different Python programming courses. Earlier generations scored below 70% on even entry-level modules and would have failed the courses were they human results. GPT-4 scored 80% or higher on the three courses and would have passed.

IMAGE BY DANIEL ALA

All the instructors interviewed in a survey conducted early in 2023 by Sam Lau and Philip Guo, researchers at the University of California at San Diego, mentioned the potential for cheating as being the primary reason they would change courses followed by the arrival of tools like ChatGPT and Copilot.

Despite the advances LLMs have made, studies have shown LLMs as large as GPT-4 still have significant limitations when it comes to understanding requests and delivering suitable code. This seems likely to reduce the potential for cheating in education and for automating away jobs in the workplace. Though the McKinsey survey of its staff found improvements on simpler tasks, time savings shrank to less than 10% on tasks that developers deemed to be more difficult, or where they were trying to use a programming framework with which they were less familiar. Sometimes, tasks took junior developers up to 10% longer with the tools than without them.

An experiment by a team working at Chang'an University, China, and Griffith University, Australia, that used CodeWars katas as a suite of tests found GPT-4 could complete more than its predecessor, but failed on all the tasks in the top three levels of the eight-level contest. The models showed difficulty with optimization and simplifying algebraic equations. In one kata, the problem asked for the total area covered by a group of rectangles. GPT-4 and its predecessor could pass simple test cases, but failed to finish before the timeout deadline on the full set of tests.

Complexity is not the only barrier. Other studies have shown that LLMs can be tripped up by the order of right and wrong answers in multiple-choice questions and the way short code snippets are presented. In one example, GPT-4 failed to notice that a string-replacement function would replace all instances of the string listed in the question, but when asked about the function, it correctly explained how it works.

Students appear to learn quickly about AI's shortcomings, despite instructors' fears of them becoming overly reliant on LLMs. In a series of interviews with students, Cynthia Zastudil and colleagues at Temple University, Philadelphia, found a common complaint lay in the black-box nature of the commercial LLMs. The students often could not determine why the AI provided them with the response it did. "When using them, I don't really fully understand what they're telling me," one student said to the interviewers.

A joint study by teams from Abilene Christian University, University College Dublin, and an Auckland, NZ-based group led by Paul Denny (see the related research article in this issue, p. 56) has informed the work of a generative-AI working group[a] set up by the organizers of the Conference on

a https://iticse23-generative-ai.github.io

Innovation and Technology in Computer Science Education (ITiCSE). The experiment uncovered types of behavior that point to students developing a healthy distrust of LLMs' answers. One behavior they called "shepherding," where the users try different variations of prompts to the AI to nudge its output closer to what they want. The other, "drifting," is when students try different variants of prompts and responses before deleting the results entirely.

"Struggling students can drift when they don't have a clue what's going on. But good students can drift too. They are testing, poking, exploring," says Brett Becker, assistant professor at University College Dublin.

The immediate question for educators as these tools evolve is whether to resist AI-enabled cheating by designing tests that LLMs today find difficult to rework the syllabus to incorporate them into courses and assessments. Instructors in the surveys by Lau and Guo split almost evenly into these two groups. Some saw a need to move to face-to-face interviews to assess skills or have students perform at least some of their assignments in a classroom environment where teachers can restrict access to LLMs, though these impose a significant overhead on instructors compared to problem sheets that today are often graded automatically.

In his talk as part of the closing keynote at the 2023 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE) in July, Denny said, "Very early on, a lot of the discussion was focused on the negative aspects. We've seen headlines about terrified professors, worrying that the students are going to be cheating endlessly with these tools. Now, the narrative has changed slightly. Maybe we should try to learn how to teach more effectively with them."

A series of papers published over the past couple of years has shown LLMs can perform tasks such as identifying programming concepts in unannotated source code, and describing how they work. This kind of work would make it easier for instructors to keep up with changes in languages and applications that industry expects and reduce reliance on potentially outdated material. Becker says similar technology would help overcome a major problem for novice students: how to interpret the often-cryptic error messages provided by compilers and other software tools.

Students are more comfortable asking an LLM for explanations than talking to a professor or teaching assistant.

The interviews by Zastudil and colleagues indicated there is a possible mismatch between student and instructor expectations that LLMs could help fill. Though both sides saw problems with tests that appear to students as being just busywork, students felt their instructors missed a more fundamental issue the AIs could address: Traditional course materials are not always helpful for understanding underlying concepts. They find they are more comfortable asking an LLM for explanations than talking to a professor or a teaching assistant, particularly for basic concepts.

Courses that incorporate LLMs already have started, though the current offerings are experimental to differing degrees. One example at the University of California at San Diego (UCSD) started in the fall of 2023. Leo Porter, a teaching professor of computer science, is developing the course and a related book with Daniel Zingaro, associate professor in computing science at the University of Toronto.

An option some educators are pursuing is to use only a specialized LLM that is prevented from supplying compilable code as part of its answers, in the hope this will prevent students just copying and pasting the output. The UCSD course will provide access to Copilot, which runs the risk of providing material that does not directly fit the course. "We need to teach students how to determine what the code does, even if that code may be more complex than what they've been taught in class," says Porter.

One possible issue with incorporating LLMs and prompt engineering into courses is the rapid pace of development that may make some techniques quickly redundant.

"We suspect the models will improve, but the way of interacting with the models won't change dramatically. We'll hopefully see correct code generated more frequently, but we don't see the models being able to generate correct code if the person writing the prompts is ambiguous about what they want," Porter notes.

Work continues to merge LLMs with other tools to improve their ability to check the correctness of their outputs by themselves. A little more than a decade ago at the Alan Turing Centenary Conference in Manchester, U.K., Tony Hoare, emeritus professor of computer science at Wolfson College, Oxford, argued for a change to the classic Turing test. This would ask the AI not to pretend to be human, but instead to reason about its own programming. As the industry struggles to determine how well LLMs can handle the logic of programs, the need for that kind of test is becoming increasingly urgent.

Further Reading

Denny, P. et al.
Computing education in the era of generative AI. *Communications 67*, 2 (2024); 56–67.

Šavelka, J., Agarwal, A., Bogart, C., and Sakr, M.
Large language models (GPT) struggle to answer multiple-choice questions about code. In *Proceedings of the 15th Intern. Conf. on Computer Supported Education*; DOI: 10.5220/0011996900003470

Lau, S. and Guo, P.J.
From "ban it till we understand it" to "resistance is futile": How university programming instructors plan to adapt as more students use AI code generation and explanation tools such as ChatGPT and GitHub Copilot. In *Proceedings of the 19th ACM Conf. on Intern. Computing Education Research* (2023); DOI: 10.1145/3568813.3600138

Deniz, B.K. et al.
Unleashing developer productivity with generative AI, McKinsey & Company (2023); https://mck.co/44uig93

Prather, J. et al.
ITiCSE working group report and student guide; https://iticse23-generative-ai.github.io

Chris Edwards is a Surrey, U.K.-based writer who reports on electronics, IT, and synthetic biology.

Society | DOI:10.1145/3638536 Gregory Mone

Virtual Reality as Therapy

Leveraging precisely designed alternate realities as therapeutic tools.

A CHILD ONCE DESCRIBED their complex regional pain syndrome (CRPS) to Stanford University professor of anesthesiology Brenda Golianu as feeling like "an orchestra of pain." Typically the condition requires treatment with multiple therapies, including nerve blocks, ketamine infusions, medications, intensive physical therapy, and psychology. Golianu recalls one example of a 12-year-old girl suffering from CRPS of the lower-right leg; she walked only with a crutch. She had been given all the treatments described here, but Golianu and her colleagues added an additional element based around a virtual reality (VR) game.

The researchers were testing VR as a way to incentivize children to move and work through difficult but beneficial physical therapy exercises and drive significant rehabilitative progress. They designed a game, FruityFeet, specifically for pediatric rehabilitation. FruityFeet situates the player in a virtual farm and instructs them to smash digital fruits and vegetables as a timer runs down.

When this particular girl took part, she entered the treatment room relying on her crutch. She then donned an immersive VR headset from the company Vive, set her walking aide aside, and proceeded to play. Sensors tracked the girl's movements, and she would receive more points if she lifted her leg higher. She was more active and likely to engage in difficult movements—a trend that emerged throughout the larger study.

"We have videos of kids coming in on crutches, stomping their feet during gameplay, then taking off the headset and using crutches again," explains Golianu. "The gains don't transfer automatically to the reality we live in, but often the kids are able to do more in virtual reality than they do in physical therapy."

This use of VR is just one example of how scientists and medical professionals are turning precisely designed alternate realities into therapeutic tools. Today, VR is helping patients overcome phobias, post-traumatic stress disorder (PTSD), a variety of musculoskeletal injuries and ailments, and more.

Until recently, headsets were too expensive, too heavy, and required cables and other hardware, according to Matthew Stoudt, CEO of AppliedVR, which has an FDA-authorized product for lower back pain called RelieVRx.

Now that is changing. The metaverse might not have arrived on schedule, but the technology improvements it generated, combined with ingenious research, have opened up incredible potential for novel treatments. "The latest headsets are marvels of display technology compared to what we had available just a couple of years ago," says psychologist Albert "Skip" Rizzo, director of Medical Virtual Reality at the University of Southern California's Institute for Creative Technologies. "They have gotten so good that we can do really valuable work in therapy. The technology has finally caught up to the original vision."

A New Treatment Option

That original vision traces back to the 1990s. Emory University psychologist Barbara Rothbaum published the first paper looking at the potential use of VR as a treatment for psychiatric disorders in 1995. Rothbaum effectively helps patients confront their fears and learn to manage them through what is known as exposure therapy. If someone has a fear of flying, for example, Rothbaum might arrange to fly with the patient to help them manage and work through their fears in real time. This technique is effective, but it is also expensive and not particularly scalable—a therapist could only do this work with so many patients, given the time commitment.

With VR, a patient who has an acute fear of flying can don an immersive headset and embark on a simulated flight. Rothbaum and her team can control the conditions precisely, too. "If my patient's not ready for turbulence, I can guarantee there won't be turbulence," she says. "We can take off and land as many times as we need, all within a five-minute therapy session right in my office."

One of her studies showed that 90% of such patients who had successful VR exposures flew on actual airplanes within a year of the treatment. These were individuals whose fear was so acute that they would drive from Atlanta to Los Angeles rather than fly, or travel by boat to Europe. "It doesn't matter if someone can get on a virtual airplane

IMAGE BY ANNA STILLS

if they can't get on a real airplane, but it really does seem to translate."

Rothbaum and her team have also done pioneering work in using VR for PTSD in collaboration with Rizzo. He notes there are now thousands of research studies documenting the value of applying these technologies to different areas of mental health and physical rehabilitation. In some cases, the effectiveness of VR is not necessarily better than traditional approaches, but the technology has the potential to draw in and reach more people.

Other examples reveal unique advantages. A study from Rothbaum, Rizzo, Cornell University psychologist JoAnn Difede, and others showed that VR exposure therapy is particularly beneficial for patients with comorbid depression and PTSD. "We could emotionally activate their trauma memories in a way that was more powerful and achieve better clinical outcomes using VR," says Rizzo.

Gamification

The use of VR for therapy is not confined to trauma and phobias. There are a number of applications similar in spirit to the FruityFeet game, as one of the core advantages of the medium is its ability to engage the user through play. At the Motor Control Lab at Virginia Commonwealth University, director James Thomas has been rigorously studying the use of VR for physical therapy, and for chronic back pain in particular. People who suffer back injuries sometimes develop a fear of moving; they worry that if they move the wrong way, they will aggravate their injury. However, Thomas explains that a lack of movement can lead to harmful changes in the muscles, ligaments, tendons, and tissues that support the spine. That is why Thomas and his team started looking at the role fear played in motivating people to avoid certain movements, and eventually reasoned that a VR game might be a good way around the problem.

Today, Thomas and his lab are refining a suite of games geared toward physical rehabilitation. One is set on a dock jutting out into a virtual lake; players need to reach forward and catch fish with a digital net as they leap out of the water—how far and how quickly they have to reach is precisely tuned to their needs. The most physically demanding

Today, VR is helping patients overcome phobias, post-traumatic stress disorder, a variety of musculoskeletal injuries and ailments, and more.

game is a virtual version of dodgeball specifically designed to encourage particular movements. A player dons the headset and enters a fully immersive competition, complete with cheering and booing fans. The avatar is scaled to match the user's body, from hip height to arm length, so movements are accurately matched. Players have to dodge virtual balls thrown their way, and the game can be adjusted to make the balls travel faster or slower, depending on where the patient is in his or her therapy.

The goal, Thomas explains, is to induce specific movements the patient has been otherwise avoiding. "It turns out that when you draw people into an environment where they are not thinking about moving, and there's clapping and booing, their natural competitive instincts emerge and they move," Thomas says. "If I launch a virtual ball into a particular part of space, I know that I can get you to move your spine a certain way, and then I can manipulate the game's controls to get you to move more or less, depending on your needs."

A recent study of 175 patients who took part in 3,500 individual testing sessions yielded encouraging results, with significant reduction in pain and fear of movement.

Technology and Outlook

In general, while they applaud the development efforts of Meta, Apple, Vive, and other large companies and manufacturers, the experts are not beholden to any particular VR headset. Rizzo would like to see the headsets become less expensive and easier for both physicians and patients to use.

AppliedVR's Stoudt echoes this point, noting that ease of use is essential for his company, given that their vision calls for patients using these headsets on themselves at home.

Display resolution is not a primary concern. Rothbaum's research has shown that even cartoonish renderings can have a huge effect. "If you have someone who is scared of snakes, if they even see a picture of a snake, they're going to feel fear," she says. "People get scared in virtual reality, and if you stay in it long enough, that fear decreases."

Currently, AppliedVR is deploying its treatments through headsets manufactured by VR leader PICO. Stoudt says PICO gave his team access to its hardware and operating system so they could optimize the design and better prepare for FDA authorization. AppliedVR also created a solution to capture breath, and measuring biodata may become increasingly popular. Generally, Stoudt believes there will be significant technological development in the years ahead.

"The next stage of the headsets is that you're going to see them become true computing platforms," Stoudt says. "It took the field a long time to get here, but now we're going to see rapid acceleration." C

Further Reading

Bordeleau, M. et al.
The use of virtual reality in back pain rehabilitation: A systematic review and meta-analysis. *The J. of Pain 23* (2022), 175–195.

Difede, J. et al.
Enhancing exposure therapy for posttraumatic stress disorder: virtual reality and imaginal exposure with a cognitive enhancer: a randomized clinical trial. *Translational Psychiatry 12*, 1 (2022), 1–9.

Griffin, A. et al.
Virtual reality in pain rehabilitation for youth with chronic pain: Pilot feasibility study. *JMIR Rehabilitation and Assistive Technologies 7*, 2 (2020), e22620.

Rizzo, A.A. and Koenig, S.
Is clinical virtual reality ready for primetime? *Neuropsychology 31*, 8 (2017), 877–899.

Gregory Mone is a Boston-based science writer and the author, with Bill Nye, of *Jack and the Geniuses: At the Bottom of the World.*

DOI:10.1145/3633585

Christopher Hoadley and Sara Vogel

Opinion

Autocorrect Is Not: People Are Multilingual and Computer Science Should Be Too

Considering the interconnection of computing and human languages.

COMPUTER SCIENCE HAS a language problem—and we are not alluding to programming languages. Many prevalent, flawed views about natural human language are limiting who is in computer science and what people can accomplish with the technology we build.

To start, computer science centers around the English language, and that produces technologies that work poorly for many people. As Manuel Pérez-Quiñones[a] points out, when developers make assumptions about English as the default language, navigating digital device interfaces can be frustrating, even for a professional computer scientist fluent in English such as Pérez-Quiñones. Poor multilingual or character-encoding support, incorrect cultural norms baked into software, and so on—these challenges confront users all over the world. Language-neutral software becomes buggy or unusable if someone inputs non-alphabetic Unicode characters (for example, Chinese ideograms), a right-to-left language, or even uses unexpected punctuation. And in the world of speech recognition, voice assistants often struggle with accented speech, proper names, and many other aspects of natural human speech.

But we believe these English-centric problems are symptomatic of a larger issue in CS: People in general, and our field specifically, think about language in narrow ways. When designers and developers think about language, it might just be as a target for localization or regionalization of software. Or, if we are multilingual, we may feel valued because our skills are commercially useful in global business dealings. We may also think about different human languages as a problem to solve in natural language processing or in information processing. We might tag prose in html files, build speech recognition or user interfaces for some global languages, and perhaps nod approvingly if a résumé crosses our desk listing languages spoken beyond English. But the typical ways we conceive of language in our work are narrow and contradict what linguists are finding about how people communicate.

Language Evolution

Language is emergent, ever-changing, and dynamic. Nation-states and technologies may have codified and attempted to standardize the lexicons, phonologies, and syntaxes of human languages—such as Hindi, Mandarin, Spanish, English, and many others, but in fact, sociolinguists underscore what people actually do with language goes far beyond what the rulebooks say. People said to speak the same language (for example, an English speaker in Louisiana and one in Scotland) may be unintelligible to each other, while people said to be speakers of different languages (for example, a Portuguese speaker in

a See https://bit.ly/48HDFOu

IMAGE BY ANDRIJ BORYS ASSOCIATES, USING SHUTTERSTOCK

Brazil and a Spanish speaker in Argentina) may find they can communicate well enough without "learning another language." The "English" used by teenagers is not the same as the "English" used by their grandparents. Recent sociolinguistics research[b] has called attention to this fact, arguing that languages are social and political categories rather than linguistic absolutes. People dynamically draw on words, grammars, and sounds they know—as well as multimodal elements like gestures, media references, clothing, and technology—to communicate their intended messages in particular contexts. Sociolinguists call this process *translanguaging*.

Language is messy and useful regardless of whether it conforms to what a grammarian may have taught in school, and if we put "academic" or "standard" English (or any other language) on a pedestal, the natural outcome of this closed-mindedness is we can exclude people who are not already part of dominant groups. Translanguaging, on the other hand, recognizes that everyone, everywhere, is using a mix of ways of communicating based on their own repertoire, and that oftentimes this crosses the boundaries of official languages with names and dictionaries. When multilingual people use words from languages other than the one the software is set to recognize, their input gets autocorrected or flagged as errors; this extends to nonstandard use of a single language, and it is even worse for people who naturally communicate in ways that defy the normative "one language at a time" model. In computer science, we need to deemphasize interfaces and natural language processing that relies on tidy official languages and correctly handle how language speakers innovate on language all the time. One reason we fail to do this is simplification; it is easier to build systems if we pretend there are a fixed set of named languages, and it certainly matches the way dictionaries, schools, and grammar textbooks are written.

But there is a deeper, and more insidious reason we pretend language is neat and tidy. Language can be used to divide groups of people in ways that keep some powerful and some powerless. Languages such as English gain prestige because their speakers have political and economic power—not because they are any better at helping people express themselves and communicate.

English, especially the kind spoken by American or British middle-class white university graduates, is in many ways a price of entry to certain groups in the community of computer scientists, whether conversing on platforms such as stackoverflow.com or participating in various workplaces. Certainly, other languages are present in different contexts; a company in Germany may embrace a mix of German and English in its work, or an IT company in Bangalore may have a dozen or more languages

b See https://bit.ly/48vWcxK

commonly used in the workplace. But people who do not speak high-status languages will tend to be marginalized in or excluded from those spaces. Whether it is a Black American woman whose Baltimore accent is perceived as "threatening" or a "disability"[c] by a manager at a tech company, or an Indian programmer whose Hindi or English reflect accents other than a middle-class education, as in all aspects of society, people may be labeled or treated as unsuitable for being programmers. Language can also be a gatekeeper to CS for young people who hope to crack into the field. For example, in the U.S., education systems hold schools accountable for the English learning of multilingual K–12 students; those students may have to prove their English competency before they can take "honors" or "enrichment" classes such as CS.

While some people argue that assimilating to the dominant languages in computing (English and the programming languages of the day) is important to build common ground, we disagree. There is no reason, for instance, a compiler should reject code that includes comments in unexpected character sets, or programmers should not be able to use programming keywords from a variety of human languages. The point is, enforced English-centric monolingualism is so common we do not even notice it, and this should change.

Given the frustration, errors, and exclusion that English dominance has caused in our field, what can we do? First, we can invest in social and technical means to increase inclusion. On the social side, we can recognize language diversity and multilingualism are the norm around the world. It is estimated there are more than 7,000 languages spoken globally. Worldwide, it is more common than not for individuals to be multilingual. We must embrace speakers of many languages in CS settings, and we must stop emphasizing English as a prerequisite for participating in computing, both for users and developers. English-only or English-dominant tech settings should immediately raise questions about who is missing and what

c See https://bit.ly/3TMMbHH

By equipping people with a new way to express ideas through code, might CS education promote metalinguistic awareness?

language is being excluded. On the technical side, we might design and test new models for more inclusive programming environments. For example, the Scratch programming environment from MIT allows children to select among dozens of languages to relabel programming blocks; we would advocate people should even be able to mix and match keywords from multiple human languages on those code blocks, or relabel blocks with language of their choosing.

Second, computer scientists, uniquely, can help invent new ways for technology to support communication and expression that were not possible before digital technology. Computer scientists communicate through code itself. As Donald Knuth described in defining the need for "literate programming,"[d] code is not merely a set of incantations to produce behavior from a computer, but a powerful medium to share certain types of ideas for other people to read and build on. Like human languages, code conveys nuance, style, and can be awkward or elegant, legible or obtuse. Like formal mathematical notation, code is a way of precisely describing concepts or propositions that is complementary to the ways we use natural human language. When we think of code as a language we can use to communicate with other computer scientists, we recognize it brings us the power to express our ideas in ways that are not only precise, but that can come to life through their execution. In our research,[e] we have seen a teacher who used the Scratch programming environment to help young multilingual people in an English-as-a-new-language course because it offered another way for them to express themselves—through a mix of home languages, multimedia, and code in addition to English—in other words, a new kind of translanguaging. Research on human multilingualism indicates bilingual learners develop a "metalinguistic awareness,"[f] that is, they can make language an object of thought and reflection, which is linked to enduring advantages bilingual learners have in later schooling. By equipping people with a new way to express ideas through code, might CS education promote metalinguistic awareness?

d See https://bit.ly/3RcrLq5

Conclusion

Thinking more about how we as computer scientists communicate and refine our ideas, we start to see ways that coding is and is not like natural human languages, and both the similarities and differences open up possibilities. As computational power for natural language processing expands, we can consider new models for coding itself that might be more flexible and evolutionary than fixed syntax and semantics—more conversational, less like a solution to an equation (see for example, the work on natural programming). And if we embrace what linguists teach us about multilingualism—that we should accept the wonderful diversity and evolution of language rather than putting language in a straightjacket—we can truly open CS as a field and our technologies themselves to the great, big, messy, and useful possibilities. C

e See https://bit.ly/3ROeB1t
f See https://bit.ly/3TFI7sS

Christopher Hoadley (tophe@buffalo.edu) is a professor in learning and instruction and a professor in computer science and engineering at the University at Buffalo, Buffalo, NY, USA.

Sara Vogel (Sara.Vogel@cuny.edu) is the computer integrated teacher education research director at the City University of New York, NY, USA.

opinion

DOI:10.1145/3625252

N. Diakopoulos, C. Trattner, D. Jannach, I. Costera Meijer, and E. Motta

Computing Ethics

Leveraging Professional Ethics for Responsible AI

Applying AI techniques to journalism.

IMAGE CREATED BY ANDRIJ BORYS ASSOCIATES WITH THE ASSISTANCE OF SHUTTERSTOCK.AI

ARTIFICIAL INTELLIGENCE (AI) is proliferating throughout society, but so too are calls for practicing *Responsible AI*.[4] The ACM Code of Ethics and Professional Conduct states computing professionals should contribute to society and human well-being (General Ethical Principle 1.1), but it can be difficult for a computer scientist to judge the impacts of a particular application in all fields. AI is influencing a range of social domains from law and medicine to journalism, government, and education. Technologists do not just need to make the technology work and scale it up, they must make it work while also being responsible for a host of societal, ethical, legal, and other human-centered concerns in these domains.[11]

There is no shortcut to becoming an expert social scientist, ethicist, or legal scholar. And there is no shortcut for collaborating with those experts and doing careful evaluations with stakeholders to understand the impacts of technology. But there is a way to at least jumpstart your knowledge and enrich your understanding of what it takes to responsibly implement technology in a social domain: domain-specific professional codes of conduct.

Professional codes of conduct are shortcuts for understanding commonly held values in a domain. They articulate general expectations for responsible practice and facilitate the understanding of issues arising around that practice. Similar to the computing field, medicine, law, architecture, journalism, and plenty of other professions have spent years working through and crystallizing what it means to act ethically in their particular domain of society.[5]

Technologists can take these codes as a starting point for informing the design, engineering, and evaluation of responsible AI systems that comply with the ACM Code of Ethics and Professional Conduct. Such codes are not checklists that will automatically make the AI (or the technologist creating that AI) "responsible." Rather, the reasoning and values embodied in such codes can help guide technologists toward more responsible choices in their practice as they develop AI-based systems for these domains.

We illustrate this idea in the context of applying AI techniques in the domain of journalism, which is familiar to most readers but differs sufficiently from academic publication in that more guidance is required than is provided by the ACM Code of Ethics. We first introduce some key professional values in journalism, elaborating a design-build-evaluate process for incorporating those values into a system, and then show how this process applies specifically to an algorithmic content-curation feed.

From Journalism Ethics to Responsible AI for Media

Professional journalism ethics has long grappled with "the responsible use of the freedom to publish."[12] Although journalism ethics is ever evolving to address new social and technical conditions, journalistic codes of conduct embed institutionalized ethical principles of proper behavior, which can guide computing professionals designing and developing new media technologies.[8] The Society of Professional Journalists (SPJ)[2] offers four main tenets: seek truth and report it, minimize harm, act independently, and be accountable and transparent (see the table).

These affirmative duties, reflecting deeply held domain values, can inform the design of an ethical media system. The accompanying figure shows how the domain-specific values in journalism can inform the design, build, and evaluation phases of engineering, show how principles and norms of responsible practitioners can be reflected in the data, algorithms, metrics, and organizational processes built into new responsible AI systems.

By first identifying key domain values, technologists can incorporate them as requirements during the design process, generating ideas for technical and organizational features that support those values.[7] Value-sensitive design methods, which "account for human values in a principled and systematic manner throughout the technical design process," can be used to help translate domain values into design features.[6]

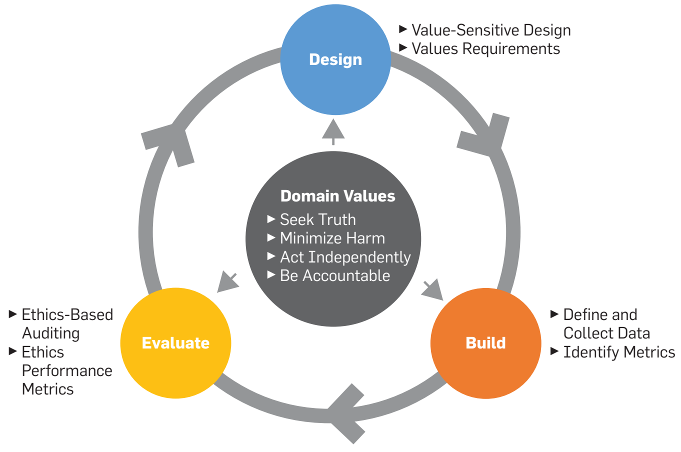
Technologists must build their systems to reflect the required values identified during design. Here, we focus on AI systems using machine learning where defining and collecting the right datasets and finding the right metrics for training and evaluation are crucial. For instance, the editorial algorithm that curates Swedish Radio's audio feeds is trained by experienced news editors rating content on whether it meets the organization's public service goals. Aligning data and metrics to values is a non-trivial challenge. Principles can be vague and multivalent, hiding potential ethical conflicts (for example, around notations of "fairness").[9] Success at the build stage requires clear definitions of values so that training data can be appropriately operationalized.

Finally, technical developers must evaluate the value alignment of the overall system. To do this, they should implement ethics-based auditing (EBA), which is a method to assess a system's "consistency with relevant principles or norms."[10] EBA supports responsible AI development and continuous improvement by evaluating a system's impact on relevant design values. There may be ethical performance metrics that can demonstrate that a value is being upheld. For instance, platforms use machine learning to detect child sexual abuse material and then report the volume of content "actioned" (for example, removed).[1] Tracking system performance against ethical performance metrics is useful internally for improving the system, and can also inform the public as part of transparency disclosures that support accountability and build trust in the system.

The four main principles of journalism ethics as described by the Society of Professional Journalists (SPJ).

Principle	Description
Seek Truth	Ensure the accuracy of published information, including providing interpretive context to avoid distortion, attributing to sources, and labeling commentary.
Minimize Harm	Interact with stakeholders (sources, subjects, the public, and impacted communities) and balance the consequences of seeking and publishing information against the harm that may cause.
Act Independently	Manage and/or disclose conflicts of interest so they can reflect the public interest without any real or perceived external influence, favoritism, or self-interest.
Be Accountable	Take responsibility for what is published and explain how you know what you know and why you are publishing something. Promptly and prominently acknowledge, respond to, and correct issues.

Responsible design and engineering of AI for media incorporating journalism-specific domain values that inform an iterative design, build, and evaluation process.



An Application to Algorithmic Content Curation

A key challenge for responsible AI is to translate abstract ethical principles into real systems. As an example, we illustrate how the SPJ principles can inform the engineering of an ethical algorithmic newsfeed for a social media platform. These values can inform the design, build, and evaluation process to ensure the principles are enacted in the sociotechnical embodiment of a feed.

To seek truth at scale in a newsfeed, designers and developers must ensure the accuracy of information in the feed. Sociotechnical design features include algorithms to filter out disinformation, amplification supervisors to manually review content receiving exceptional levels of attention, and content labels for sources and opinions. To build such features, data will often need to be defined and collected to train models. For instance, fact-checking specialists might annotate content for training and improving models to reduce misinformation. For model evaluation, feed operators should track feed-quality metrics by sampling and evaluating quality both automatically and manually using a systematized rubric.

Content moderation is required to help minimaze harm that content can cause to individuals. Data scientists might design and develop classifiers that automatically detect content about non-public victims of crime, abuse, bullying, or violations of privacy and block its amplification. Similar classifiers could filter out content for vulnerable people to avoid, for example, content reinforcing depression, eating disorders, or self-harm, and then evaluate the impact of that filtering. Harms to society, such as affective polarization, could be evaluated with key metrics (for example, engagement across political lines) to adjust the system over longer timeframes.

Independence in the feed demands conflicts of interest are managed and disclosed so the public interest is prioritized. Platforms could develop a database of sources, how they are funded (for example, commercial, non-profit, sponsored, and so forth), who owns them (for example, corporations, hedge funds, foreign entities), whether they are paid by the feed operator, and so on. To keep the database updated, new sources could be automatically identified based on feed exposure, with details filled in by trained curators. Source information could be disclosed in the user interface (for example, with labels) to clarify where information comes from and the relationship between the feed operator and the source. Evaluations of the newsfeed could assess exposure to different types of sources, such as foreign media.

Upholding the principle of accountability requires the feed algorithm provide transparency including things like datasheets and thick descriptions of system processes.[3] Periodic disclosures of internal evaluations (for example, ethics-based audits) could clarify how the feed operator is upholding the values of seeking truth, minimizing harm, and maintaining independence with key performance indicators that can identify lapses (for example, increased exposure to disinformation). Ultimately the company is accountable for how the system functions and creates a user experience, even if they do not fully control all the components that contribute to that experience. "Algorithm explainer" roles could be created with full access to system information and responsibility to explain system behavior in case of errors or malfunctions. These technical workers would release public reports or respond directly to individuals seeking redress.

Technical experts have an essential role to play in designing and engineering AI systems to be socially responsible.

Conclusion

Technical experts have an essential role to play in designing and engineering AI systems to be socially responsible. While technologists cannot all be experts in social science, ethics, or law, they can effectively leverage institutionalized domain values as generative design tools for feature suggestions, and as guides for developing and evaluating the value-aligned implementation of a system. Domain values should be seen as a way to coarsely aim the process in the right direction, but it is important to emphasize that technologists will also need to iterate on systems through deep human-centered work with various stakeholders in the domain. And as jurisdictions around the world move to regulate AI, such as in the DSA and AI Acts in the E.U., the same techniques for aligning with domain values will facilitate developing systems that are adherent to broader social values, such as fundamental rights.

By centering the values of the domain present in professional codes of conduct we can leverage the received wisdom of thousands of professionals who have already worked through some of the stickiest of ethical issues and ambiguities of responsible practice. Just as we have demonstrated here for journalism, other domains (for example, law, medicine) would similarly benefit by leveraging domain-specific ethics codes in designing and aligning responsible AI systems, including for fine-tuning some of the latest generative AI models, such as those powering ChatGPT, to align them with expectations of responsible behavior in particular domains. Let's be sure that as designers and engineers we incorporate such wisdom, and build our technologies to aspire to and embody domain values of responsible practice and of The ACM Code of Ethics and Professional Conduct. C

References

1. *Automated Content Moderation: A Primer*. Stanford Cyber Policy Center, (2022); https://bit.ly/3LT2qyr.
2. Brown, F. *Media Ethics: A Guide for Professional Conduct. Society of Professional Journalists Foundation*. SPJ Code of Ethics (2020); https://www.spj.org/ethicscode.asp.
3. Diakopoulos, N. and Koliska, M. Algorithmic transparency in the news media. *Digital Journalism 5*, (2016), 7.
4. Dignum, V. Responsibility and artificial intelligence. *Oxford Handbook of Ethics and AI*. M. Dubber and F. Pasquale, (Eds). Sunit Das. 2020.
5. Frankel, M.S. Professional codes: Why, how, and with what impact? *J. Business Ethics 8*, 2–3 (1989); 10.1007/bf00382575
6. Friedman, B. and Hendry, D.G. *Value Sensitive Design: Shaping Technology with Moral Imagination*. MIT Press, 2019.
7. IEEE Standard Model Process for Addressing Ethical Concerns during System Design. IEEE Std 7000-2021. (2021), 1–82; 10.1109/ieeestd.2021.9536679
8. McBride, K. and Rosenstiel, T. *The New Ethics of Journalism: Principles for the 21st Century*. CQ Press. 2014; *Ethics for Digital Journalists: Emerging Best Practices*. L. Zion and D. Craig, (eds). Routledge, 2015.
9. Mittelstadt, B. Principles alone cannot guarantee ethical AI. *Nature Machine Intelligence 1*, 11 (2019), 501–507.
10. Mökander, J. et al. Ethics-based auditing of automated decision-making systems: Nature, scope, and limitations. *Science and Engineering Ethics 27, 4* (2021); 10.1007/s11948-021-00319-4
11. Trattner, C. et al. Responsible media technology and AI: Challenges and research directions. *AI and Ethics*. (2021).
12. Ward, S. *Disrupting Journalism Ethics (Disruptions)*. Taylor and Francis, 2019.

Nicholas Diakopoulos (nad@northwestern.edu) is a professor of communication studies and computer science, Northwestern University, Evanston, IL, USA.

Christoph Trattner (christoph.trattner@uib.no) is a professor of information science and media studies, University of Bergen, Bergen, Norway.

Dietmar Jannach (Dietmar.Jannach@aau.at) is a professor of computer science, University of Klagenfurt, Wörthersee. Austria.

Irene Costera Meijer (icostera.meijer@vu.nl) is a professor of journalism studies, Vrije University Amsterdam, Netherlands.

Enrico Motta (Enrico.Motta@open.ac.uk) is a professor of knowledge technologies, Open University, London, U.K.

This work is funded in part by MediaFutures partners and the Research Council of Norway (grant number 309339).

opinion

DOI:10.1145/3634901

Thomas Haigh

Historical Reflections
How the AI Boom Went Bust

Fallout from an exploding bubble of hype triggered the real AI Winter in the late 1980s.

In my last two columns (June 2023 and December 2023) I followed the history of artificial intelligence (AI) as an intellectual brand and subfield of computer science, from its creation in 1955 through to the end of the 1970s. While acknowledging that AI faced high-profile skepticism from the mid-1960s onward, I argued the 1970s were a time of steady growth for the AI research community. Contrary to popular belief, the "first AI winter" of the 1970s never happened. The 1980s, in contrast, saw the rapid inflation of a government-funded AI bubble centered on the expert system aproach, the popping of which began the real AI winter: a two-decade slump. I will tell that story here, but first I want to say something about how the maturation of AI played out in textbooks and in the computer science curriculum.

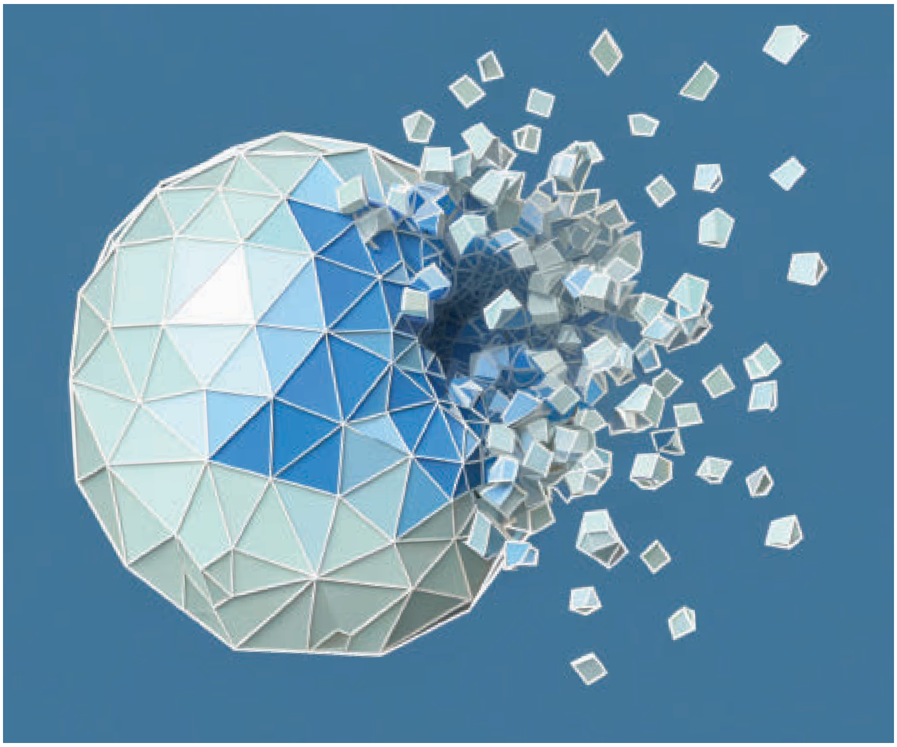


IMAGE CREATED BY ANDRIJ BORYS ASSOCIATES WITH THE ASSISTANCE OF SHUTTERSTOCK.AI

AI in the Curriculum

AI researchers dominated the first 10 years of ACM's A.M Turing Award, suggesting AI initially occupied the intellectual high ground of computer science. Looking at the computer science curriculum hints at a different story, in which AI moved from a marginal subject in the initial degree programs of 1960s to a core field by the end of the 1980s. The history of computer science education remains understudied, but we can get a fuzzy sense of developments by looking at the evolution of ACM's recommended curricula.[2] These recommendations have a complex relationship to actual practice. Likely they were most closely followed by mid-tier institutions, able to hire across a range of specialties but less likely than Stanford or MIT to have the confidence to build their own unique models around in-house expertise. The first ACM model curriculum, from 1968, described 22 undergraduate courses, including one on "artificial intelligence and heuristic programming." As an advanced "methodology" elective this was recommended only for masters' students and for undergraduates pursuing a concentration in theoretical computer science (one of six sample concentrations).[a] The course description suggested a lack of faith in the intellectual maturity of AI: "As this course is essentially descriptive, it might well be taught by surveying various cases of accomplishment in the areas under study."

A decade later, the Curriculum '78 working group recommended an elective covering "basic concepts and techniques," in AI with knowledge representation, search, and system

a https://bit.ly/47b8cmu

architecture as the main topics.[b] It also recommended coverage of LISP, an AI-focused language, in the core course on data structures and algorithms. AI was edging toward the mainstream of a rapidly expanding major. Some 15,121 bachelor's degrees in computer science were awarded in the U.S. in 1980–1981 versus just 2,388 a decade earlier.[c]

In 1988, an ACM task force chaired by Peter Denning released a report on the computer science curriculum, which identified artificial intelligence and robotics as one of nine core areas.[d] ACM's next detailed model curriculum, released in 1991 in collaboration with the IEEE Computer Society, codified AI and robotics as one of 10 top-level subject areas to be covered by all students (albeit with just nine lecture hours, on a par with databases, human-computer interaction, and numerical computation).[e]

The gradual mainstreaming of AI in the computer science curriculum was already apparent in the early 1990s when I studied computer science. The University of Manchester offered specialized AI undergraduate and graduate courses, supported by a team of four AI faculty, several allied faculty focused on formal methods and logic, and a cluster of postdocs and funded Ph.D. students. None of them won ACM A.M. Turing Awards or received gigantic grants, but the group's professor had been a student of Herb Simon, and I had the sense of being competently inducted into a well-established body of techniques. Jumping forward to the present day, the Association for the Advancement of Artificial Intelligence has joined ACM and the IEEE Computer Society as a third partner in the latest computer science curriculum update.

The growth of undergraduate AI courses reflected the new availability of textbooks, replacing teaching anthologies with more coherent volumes that attempted to draw out principles and theories. I identified seven AI textbooks published from 1971 to 1977.[1,7,8,11,12,16] The books reflected and reinforced the exceptional ability of MIT and Stanford to shape the AI brand by determining the topics and approaches to be taught elsewhere. Their eight authors all held degrees from MIT or Stanford; three had earned Ph.D.s under the direction of Marvin Minsky. At the time their books were published, four authors worked at the Stanford Research Institute (which had by then separated from the university). The most widely adopted of the early textbooks was published in 1977 by Patrick Henry Winston, the longtime director of MIT's AI lab.[18] Fifteen years later, as a student, I was assigned an updated edition. Winston's first serious competition came from Nils Nilsson, an SRI researcher and eventual Stanford professor, whose text *Principles of Artificial Intelligence* appeared in 1980. Elaine Rich was a recent Ph.D. graduate of Carnegie Mellon when her textbook appeared in 1983. Through several editions with new coauthors it became the main rival to Winston's book.

Early AI had imagined general-purpose reasoning engines driven by collections of individual facts.

The major textbooks of the era dealt entirely with symbolic approaches to AI, neural networks having been purged from the mainstream of computer science. Winston never mentioned connectionist approaches even though his book reflected his specialization in machine learning and computer vision, two areas that have today become synonymous with neutral networks. Rich dismissed connectionism in two sentences: "Although there have been many attempts to build learning programs starting with a random network, none of them have met with any degree of success. For this reason, we will not discuss this approach any further here."[13] The techniques we practiced in Manchester were dominated by symbolic AI and expert systems, though we were told about statistically based techniques for natural language parsing. I took four AI courses without learning anything about neural networks or genetic algorithms, which were confined to a final-year elective."

From Reasoning to Knowledge

Insider histories of AI agree the crucial intellectual shift of the late 1960s and 1970s was a shift of emphasis away from the hunt for powerful reasoning mechanisms and toward more effective ways of representing knowledge. As Rich wrote in her 1983 textbook, "one of the few hard and fast results to come out of the first 20 years of A.I. research is that *intelligence requires knowledge*."[13]

Early AI theorists had imagined general-purpose reasoning engines driven by collections of individual facts. But researchers concluded that a vast amount of background knowledge was needed to accomplish apparently basic tasks, such as correctly parsing out the verbs and nouns in a sentence or understanding a simple dialogue. From 1974 onward, Minsky talked about the idea of using frames to represent types of objects and events in hierarchies. Frames combined procedures, default values, and facts. The approach strongly paralleled object-oriented programming, developed around the same time. I remember learning about ideas such as inheritance and subclassing in my AI classes rather than my programming courses.

Under Minsky's direction, a generation of researchers trained at MIT worked on microworlds. Searching through a tree of possible states for a desired goal was still the central mechanism in AI, but any general-purpose AI system would confront so many possible sequences of actions that a computer would run out of time and memory long before settling on a reasonable decision. Restricting the complexity of the modeled world made things tractable. The most famous of these systems, and one featured prominently in AI textbooks for decades to come, was SHRDLU, created by Terry Winograd for his 1971 thesis. Winograd's thesis created such a stir that it was published the next year as a full issue of the journal *Cognitive Psychology*.

SHRDLU was described as a program for understanding natural language. It accepted English language questions and commands submitted

b https://bit.ly/48i87yy
c https://bit.ly/3RsTP7v
d https://bit.ly/48i8clQ
e https://bit.ly/3GPv40e

via teletype and typed out responses to the user. The microworld it simulated was a table littered with blocks of different shapes, sizes, and colors that could be placed on top of each other by an imaginary robot arm. The computer's console display rendered the block world in wireframe graphics. The extreme simplicity of the simulated world let Winograd integrate parsing and modeling, implementing each verb as a subroutine. In a lengthy dialogue, SHRDLU responded politely and correctly to questions such as "Is there anything which is bigger than every pyramid but is not as wide as the thing that supports it?" It could answer questions about its own actions, flag ambiguities in questions, and correctly resolve pronouns.

For decades to come, anyone studying AI was likely to learn about SHRDLU and to read an extract from the famous dialogue between Winograd and his creation. But SHRDLU also encapsulated the limitations of traditional AI. While textbook authors looked for unifying principles, most notably search techniques and knowledge representation, the continuing intractability of the key problems addressed by AI researchers meant that textbooks consisted mostly of detailed descriptions of highly specialized systems, few of which were ever applied beyond carefully chosen demonstration problems. SHRDLU's dazzling demonstration script exemplified this, by giving the illusion of having achieved far more than it actually had. As Michael Wooldridge put it, researchers expected "that the techniques it embodied might provide a route to more general natural-language understanding systems, but this hope was not realized."[19] Winograd later became a critic of his own early work, saying the impressive dialogue had been carefully scripted and that even within its limited domain his program was never robust enough to work reliably.[17] He turned away from AI research, becoming instead a theorist of software design and human-computer interaction.

Expert Systems

Although theoretical computer science had displaced AI as the most fertile ground for Turing Awards, the prize committee returned to the field in 1994 to honor a second generation of AI researchers with awards to Edward A. Feigenbaum and Raj Reddy. Reddy, a pillar of Carnegie Mellon's AI program, had built startlingly capable speech recognition systems, based on a model of separate processes using a blackboard to exchange information.

Feigenbaum's Turing Award profile introduces him as the "father of expert systems," a brand that in the 1980s was often promoted as a less controversial alternative to artificial intelligence.[f] Feigenbaum, a Stanford professor and student of Herb Simon, launched the Heuristic Programming Project in the late 1960s. Like Minsky and many other AI researchers, Feigenbaum emphasized the importance of encoding knowledge. But his focus was on automating the work of human experts, initially scientists and doctors. His first system, Dendral, was developed in collaboration with Nobel prize-winning scientist Joshua Lederberg to guess the structure of chemical compounds when fed with formulae and mass spectrogram data.

Feigenbaum and his graduate students went on to develop many other expert systems, including Mycin, a tool for the diagnosis of blood infections. This led in turn to Emycin, which extracted the core reasoning part of Mycin to create a shell that could be loaded with rules encoding expert knowledge from other domains. Distilling expert knowledge into rules was the work of skilled *knowledge engineers*. First they interviewed experts, then they formulated candidate rules. Loading these rules into an inference engine such as Emycin and running them against test cases let the knowledge engineer see where it made mistakes, then explore the chain of rules that led to the error and consult the expert to determine what needed to be changed. Soon, claimed Feigenbaum, the system works as well as a human expert. Recent AI approaches involve training systems automatically against huge volumes of data. Feigenbaum insisted (and still insists) that expert systems need only a few hundred carefully chosen rules to equal the decision-making ability of high-functioning professionals.

f https://bit.ly/4aFIVEe

Replacing scarce and expensive human experts with packages of rules was a compelling pitch.

When the Boom Was On

Replacing scarce and expensive human experts with packages of rules was a compelling pitch. Expert systems launched a wave of private investment in AI, with startup companies selling software tools, system-building services, application-specific services, and implementations of the Lisp and Prolog programming languages. Apparent proof that expert systems could save money in practice was provided by the XCON system designed by Carnegie Mellon professor John McDermott to automate the translation of customer requirements for DEC's VAX computer systems into manufacturing configuration. The initial release condensed expert knowledge into 480 configuration rules, implemented using a specialized language developed with DARPA funds.[g] Almost every textbook or magazine discussion of expert systems explained that XCON had eliminated a lengthy review and testing process to shorten VAX delivery times by months. DEC boasted that XCON and a related system saved more than $40 million a year.

Startup companies proliferated. MIT alone spawned two companies selling expensive workstations with custom processors designed to run Lisp efficiently. The career of Peter Hart, who I mentioned earlier as one of the creators of the A* search algorithm, captures the ups and downs of AI. When ARPA money for SRI's robot project dried up, he made a name for himself in expert systems research with the Prospector geological sys-

g https://bit.ly/3vaEalN

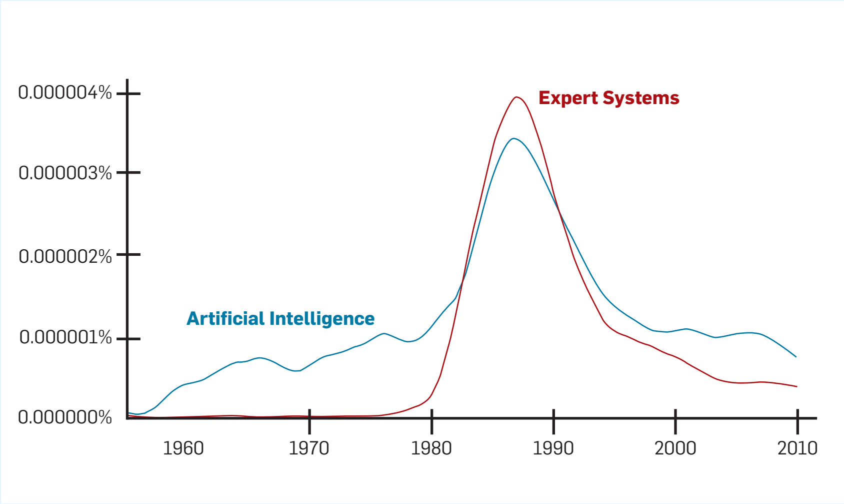
tem, then ran an AI lab for Schlumberger Ltd., and in 1983 partnered with fellow SRI veteran Richard Duda to start an expert system services company named Syntelligence. McDermott too founded a company, the Carnegie Group. Feigenbaum himself cofounded three companies. As Hart recalled the era, "new expert systems companies were being formed at the rate of what seemed like one a week."[6]

Like the earlier waves of AI enthusiasm, the new boom had a lot to do with government spending. This time it was fear of Japan, rather than the USSR, that unlocked the public purse. Japan's commitment to a human-centered approach to computing in its high-profile Fifth Generation Project included an effort to create natural language interfaces. Feigenbaum led a hugely successful campaign to present this as a major economic threat to the U.S., warning that only massive public investment in expert systems could prevent Japan overtaking the U.S. in computing just as it had in television and motorcycle manufacturing. Feigenbaum called for "a national plan of action, a kind of space shuttle program for the knowledge systems of the future."[4,5]

Politicians attempted to capitalize on a widespread belief that a microcomputer revolution was about to usher in a post-industrial society or information society in which leadership in computer technology would be much more important than traditional manufacturing industry as a contributor to national success. Britain launched the Alvey project and Europe established the transnational ESPRIT research initiative.

The most ambitious project of the era was Cyc, led by former Stanford and Carnegie Mellon faculty member Doug Lenat, a specialist in systems that made discoveries. Whereas expert systems aimed to capture knowledge in extremely narrow domains, Lenat dreamed of equipping an AI logic engine with an everyday knowledge base broad enough that it could add automatically to its base of facts and even invent new heuristics. That would take a lot of knowledge: the Cyc name came from "encyclopedia." Lenat estimated codifying an encyclopedia worth of knowledge into a gigantic semantic network would take approximately 2,000 years of person-effort. After that, the system would know enough to assimilate everything else by reading books and newspapers. Starting in 1983, Lenat got 400 researchers and more than $500 million from the Microelectronics and Computer Technology Corporation (MCC), an industrial consortium sponsored by the U.S. government to counter the Japanese threat.

Google's Ngram Viewer, based on a large English text corpus, suggests discussion of AI surged in the 1980s, driven by interest in expert systems, but declined throughout the two-decade "AI winter" that followed. Source: https://bit.ly/3ROXigO



The AI Winter

DARPA jumped back into AI in a big way in 1983 with its Strategic Computing Initiative, the story of which was told in a fascinating book by Alex Roland and Philip Shiman.[14] The program was sold to Congress with promises of direct military applications, and rested on the assumption that existing approaches to expert systems, natural language understanding, and vision were ready for large-scale application once computer hardware improved (something the program aimed to accelerate with support for research on massively parallel supercomputers, microelectronics, and prototyping). These technologies would be integrated into military systems, with self-driving vehicles selected as a test case.

In 1984, a distinguished panel convened at the annual meeting of the American Association for the Advancement of Artificial Intelligence. The conference was starting to feel like a trade show. Expert system startups were mushrooming, large corporations were rushing to establish AI groups, government money was flooding in, and a frenzied job market ensured lucrative employment for anyone who could claim a few months of AI experience. Yet introducing the panel on "The Dark Ages of AI," Yale professor Drew McDermott warned of a feeling of "deep unease" that excessively high expectations for AI "will eventually result in disaster." To sketch a worst-case scenario," continued McDermott, "suppose that five years from now the strategic computing initiative collapses miserably as autonomous vehicles fail to roll. The Fifth Generation turns out not to go anywhere, and the Japanese government immediately gets out of computing. Every startup company fails. Texas Instruments and Schlumberger and all other companies lose interest. And there's a big backlash so that you can't get money for anything connected with AI. Everybody hurriedly changes the names of their research projects to something else."[10]

McDermott noted this "unlikely" scenario was so apocalyptic that it was "called the 'AI Winter' by some," in reference to scientific debate over the prospect that nuclear war would throw enough soot into the atmosphere to trigger devastating global cooling in a nuclear winter. Super-power diplomacy staved off the nuclear winter but by the end of the decade the AI apocalypse was taking place just as described.

At DARPA, for example, speech recognition work progressed well but other strategic computing projects disappointed. Reagan-era budget cuts also contributed to a scaling back of effort and expectations. At the end of 1987, DARPA abandoned the flagship effort to build an autonomous land vehicle (though work it had funded at Carnegie Mellon's Navlab provided an important foundation for later developments). DARPA's leadership "elected simply to sweep Strategic Computing under the carpet and redirect computer research toward the 'grand challenges' of high-performance computing. Numerical processing replaced logical processing as the defining goal."[14]

The AI Winter is clearly visible on Google's Ngram chart in this column. Discussion of AI grew steadily through the 1970s before spiking in the 1980s. This was tied to an explosion of discourse about expert systems, a phrase that at its peak in the late 1980s was just as common as "artificial intelligence" itself. Both fell precipitously during the 1990s. By 2010, references to AI were coming less than one-third as often as they had at the peak and the rate was still falling.

Discussion of expert systems dropped more rapidly, reflecting the collapse of the short-lived industry. Comparing the expert system story with the approximately contemporaneous commercialization of relational database management systems is instructive. Both began with bold ideas of disputed practicality, followed by impressively engineered prototype systems produced in industrial and academic labs. Both technologies were recognized with Turing Awards, and both were commercialized as software platforms marketed by startups with close connections to universities. In the case of relational database management systems, the crucial work was done by IBM Research and at the University of California, Berkeley. Relational database management companies thrived, turning their products into universal infrastructures for corporate data. The best known of them, Oracle, is among the world's most successful businesses.

In contrast, the market for expert system software proved unsustainable because most companies struggled to build the in-house skills needed to use them effectively. Companies that had set up AI groups and purchased expert system software discovered systems designed to automate expertise required them to hire new experts to maintain them. By 1989, DEC had 59 technical staff members assigned to maintain the infrastructure and base of rules for its internal expert systems, which remained the most widely publicized application of AI.[h] Few companies could sustain such investments, particularly as a shortage of AI specialists had driven up wages.

Lenat's grand vision for Cyc did not materialize either, in part because developing a single consistent knowledge base proved impossible, but the project continued. In 1994, as the MCC began to implode, the Cyc project was transferred to a private company that continues to develop and license Cyc. It has now grown to a collection of 30 million rules.[3,9]

The AI Winter extended to the Turing Awards. In the eyes of 16 successive selection committees, the field of AI failed to produce anything between 1995 and 2010 to match the advances in areas such as databases, cryptography, networking, programming, and complexity theory that were honored with awards.

Broad-based and sustained as this decline in discussion of AI was, it may not reflect experiences outside the U.S. and U.K. and likely understates the resilience of AI as an area of computer science teaching and research. In South Korea, for example, AI publications and funding rose steadily in the late 1980s and early 1990s.[15] Because conventional histories of AI (at least those in English) have constructed AI as an almost entirely Anglo-American project, this and other aspects of its history must be reassessed when that focus eventually broadens.

AI returned to primetime in the 2010s with the dramatic revival of interest in connectionist approaches centered on deep learning systems. The effort began in the 1980s but, because AI had been redefined around symbolic approaches, was pursued under other brands, such as machine learning and pattern recognition. Only in the last few years has the AI brand itself been flipped to refer primarily to deep learning and generative systems. In my next column I will be telling that story and looking at differences and parallels between our current wave of AI hype and the booms and busts of years gone by.

h https://bit.ly/41v5Wp4

References

1. Duda, R. and Hart, P. *Pattern Recognition and Scene Analysis*. Wiley, New York, 1973.
2. Dziallas, S. and Fincher, S. The history and purpose of computing curricula (1960s–2000s). In *Communities of Computing: Computer Science and Society in the ACM*, T.J. Misa, Ed. Morgan & Claypool (2017).
3. Ekbia, H.R. *Artificial Dreams: The Quest for Non-Biological Intelligence*. Cambridge University Press, New York, 2008.
4. Feigenbaum, E.A. and McCorduck, P. *The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World*. Reading, MA, 1983.
5. Garvey, C. Artificial intelligence and Japan's Fifth Generation: The information society, neoliberalism, and alternative modernities. *Pacific Historical Review 88*, 4 (2019).
6. Hart, P.E. An artificial intelligence odyssey: From the research lab to the real world. *IEEE Annals of the History of Computing 44*, 1, (Jan.–Mar. 2022).
7. Hunt, E.B. *Artificial Intelligence*. Academic Press, New York, 1975.
8. Jackson, P.C. *Introduction to Artificial Intelligence*. Petrocelli Books, New York, 1974.
9. Lenat, D. Creating a 30-million-rule system: MCC and Cycorp. *IEEE Annals of the History of Computing 44*, 1 (Jan.–Mar. 2022).
10. McDermott, D. et al. The dark ages of AI: A panel discussion at AAAI-84. *AI Magazine 6*, 3 (1985).
11. Nilsson, N.J. *Problem-Solving Methods in Artificial Intelligence*. McGraw-Hill, New York, 1971.
12. Raphael, B. *The Thinking Computer: Mind Inside Matter*. W. H. Freeman & Company, San Francisco, CA, 1976.
13. Rich, E. *Artificial Intelligence*. McGraw-Hill, New York, 1983.
14. Roland, A. and Shiman, P. *Strategic Computing: DARPA and the Quest for Machine Intelligence*. MIT Press, Cambridge, MA, 2002.
15. Shin, Y. Hangul and the "spring" of artificial intelligence research in South Korea. *Technology's Stories 6*, 1 (Mar. 2018).
16. Slagel, J.R. *Artificial Intelligence: The Heuristic Programming Approach*. McGraw-Hill, New York, 1971.
17. Winograd, T. *Oral History Interview by Arthur L. Norberg*. Charles Babbage Institute. (1991); https://hdl.handle.net/11299/107717
18. Winston, P. *Artificial Intelligence*. Addison-Wesley, Reading, MA, 1977.
19. Wooldridge, M. *A Brief History of Artificial Intelligence: What It Is, Where We Are, and Where We Are Going*. Faltiron Books, New York, 2021.

Thomas Haigh (thomas.haigh@gmail.com) is a professor of history at the University of Wisconsin—Milwaukee, WI, USA, and a Comenius visiting professor at Siegen University, Germany.

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)-Project-ID 262513311-SFB 1187 Media of Cooperation.

DOI:10.1145/3633288

George V. Neville-Neil

Article development led by acmqueue
queue.acm.org

Kode Vicious
Dear Diary

On keeping a laboratory notebook.

Dear KV,

I recently joined a medical company as a data scientist—crunching numbers on its latest drugs—and one thing I have noticed is that biologists and other noncomputer people I work with keep notebooks. Every single experiment or idea is carefully recorded in a physical notebook that is then checked by a co-worker. The process seems laborious, but I am told it is required for their work. I cannot remember ever having to record experiments for my computer science courses in college, admittedly more than a few years ago, but I know you have written about keeping a debug log. Is that the same thing?

Noted

Dear Noted,

Congratulations on the new job: Yes, you have a good memory. I have written about using a debug log to figure out problems and apply the scientific method to debugging. While a debug log is helpful, it is not the same thing as a laboratory notebook. Did you know the first actual bug was a moth found in a relay of the Harvard Mark II computer in the late 1940s by doctor (and, eventually, Rear Admiral) Grace Murray Hopper? She not only found the bug, but also taped it into the debug log. (You can find the log entry and image online at https://americanhistory.si.edu/collections/search/object/nmah_334663.) If only all our bugs were large enough to see!

I have to say I find it surprising a data scientist would not know about laboratory notebooks. In the physical sciences

IMAGE BY OLGA STREL

such as physics and chemistry, as well as in the medical sciences, such notebooks are required. In fact, undergraduates in those classes must turn them in for grading as part of their studies. Most lab notebooks remain on paper, although there are also expensive online systems for keeping lab notes. The online systems are meant to protect companies from predatory patent trolls and to provide a digital way of proving their invention, whatever that may be, was first.

We in computer science can learn from the long-standing processes in other sciences, and we should get ourselves up to date with the 18th century. Maintaining a proper lab notebook has been a thing in the noncomputer sciences for several hundred years, and it's time computer science earned its science badge by going back to the future.

Most people who spend their days in front of computers would probably balk at keeping a paper notebook of experiments instead of a more convenient electronic form. Let's see how we could keep an electronic laboratory notebook without paying through the nose.

Laboratory notebooks are different from other notebooks and logs in a few ways. A laboratory notebook should contain a series of experiments and experimental notes that work out a hypothesis. The hypothesis does not need to be grand: "Light is both a wave and a particle, and, therefore, gravitation will divert the path of light." It can be much like the debug log entries mentioned previously, or it can be a hypothesis about a particular change to the code: "Substituting a Fortran mathematical routine in this set of calculations for its C++ equivalent will reduce the time of the computation by a factor of 2." The hypothesis is something that can be tested and should be stated simply enough that anyone else conversant in our scientific endeavor can understand it.

Each experiment should have a title and an unambiguous date. KV prefers Day-Month Name-Year, since even a non-English speaker can figure out the

Example Notebook Entry

1 Sep 2023 Getting the time in user space: in_progress:

HYPOTHESIS
What are you trying to show? Be specific about what you are measuring: time, transactions, size of data, latency, and so forth.

HOW
(Procedures, calculations, equipment)
- **The details of How**
- **Equipment: CPU, memory, disk, network features, and base performance characteristics**
- **Software: Name, version, options**
- **Scaffolding scripts, executables: Location, name, command arguments passed**
- **Commands typed: Use the `script(1)` command before you start any interactive session that will run test or other measurement code. This command captures all commands typed and their output for later use.**

OBSERVATIONS
- **Describe all that happens (planned or unplanned) during the experiment in narrative text.**
- **Raw experimental data: Tables and other data that are small enough to fit directly into the notes may be placed in this section.**
- **Large output files: Point to the files, signed and stored in a repo, that contain any relevant output.**

Org mode has a way to strike out text, C-c C-x C-f +, which inserts plus-mark characters that make a strikeout as in the following line:

~~+ This idea did not work out and therefore is struck out .+~~

In a laboratory notebook we NEVER remove ideas; we leave them and strike them out. Yes, we can recover text via the version-control system but that is not sufficient for our purpose. We need to have the strikeouts remain to work as a real lab notebook.

DATA ANALYSIS
Processing of raw data, graphs, interpretations
- **Summarize your interpretation of the results of the experiment in narrative text.**
- **Summary tables, graphs, images, and other items may be placed directly into the notes.**
- **Large tables, graphs, images, and so forth: Point to the files, signed and stored in a repo, that contain any relevant output.**

IDEAS FOR FUTURE EXPERIMENTS
A list of further experiments you believe would tease out new information.

thing in the middle is not a numerical day or a year. KV also keeps a state variable associated with the entry: Is it not started or is it in progress (lots of head banging on the desk trying to build and run the experiment)? Once complete, it can be marked COMPLETE or FAILED. More experiments FAIL than COMPLETE for KV, but maybe you are better than that. The goal is not to have a perfect record of COMPLETEs, but to have that *one* experiment that satisfies the hypothesis.

Once you have your hypothesis, you must test it, but how? The next section of the notebook lays out the procedures, calculations, equipment, software libraries, and everything that is required for the experiment. The How section should include *everything* about the experiment you can think of. Many experimenters have failed because they did not include everything they used, and so they could not tell that, for example, running the experiment with different pieces of software on the same system would affect the outcome, or even mask an effect they were looking for.

Write down everything you can think of, and then ask a colleague if you have left out anything. In the physical sciences, this type of pair verification is common.

Now it is time to run your experiment and record your observations, which is your next section. Much as you did in the How section, record everything you see (hear, taste, smell—whatever senses you can bring to bear on the problem will later serve you well when you do the analysis).

Note that observations are just that: Just observe, be as one with the experiment, be the observer. Later, you can analyze.

Eventually the observations will, ideally, complete, subject to the Turing limit, and it will be time to analyze the observations. This is where you summarize everything you thought you observed and try to discover if you have proven or disproven your hypothesis. You remember the hypothesis, right? It is at the top of the entry! Go reread it. Did the work you did prove or disprove anything? If you had false assumptions or observations, cross them out (there are convenient type fonts for this now) but leave them in the entry.

Finally, the experiment may have given you ideas for further experiments. As that great Irish scientist and comedian Dara Ó Briain said: "Science knows it doesn't know everything; otherwise, it'd stop." Write down what you might want to figure out next.

To be presumptuous, I have included a template page from one of my own lab notebooks in this column. It should be used as a starting point and not as gospel, because the gospel according to KV would contain a lot of blaspheming.

KV

Recommended resources

For an excellent introduction to keeping a lab notebook, check out Howard M. Kanare's *Writing the Laboratory Notebook* from 1985.

Here is the github repo for notebooks I set up: https://github.com/gvnn3/labnotebook.

And here is the Org Mode Notebook file: https://raw.githubusercontent.com/gvnn3/labnotebook/main/samples/notebook.org.

Related articles on queue.acm.org

Gettin' Your Head Straight
Kode Vicious
https://queue.acm.org/detail.cfm?id=1281885

Some Rules and Restrictions May Apply
Stan Kelly-Bootle
https://queue.acm.org/detail.cfm?id=1661787

Kode Vicious Cycles On
Kode Vicious
https://queue.acm.org/detail.cfm?id=1080870

George V. Neville-Neil (kv@acm.org) is the proprietor of Neville-Neil Consulting, Brooklyn, NY, USA, and co-chair of the *ACM Queue* editorial board. He works on networking and operating systems code for fun and profit, teaches courses on various programming-related subjects, and encourages your comments, quips, and code snips pertaining to his *Communications* column.

opinion

DOI:10.1145/3624729

Rahul Simha, Amruth N. Kumar, and Rajendra K. Raj

Opinion
Undergraduate Computer Science Curricula

First-job readiness versus long-term career preparation.

There can be many conflicting goals for the design of a computer science curriculum, including: immediate employability in industry, preparation for long-term success in an ever-changing discipline, and preparation for graduate (that is, post-graduate) study. Emphasis on immediate employability may lead to prioritizing current tools and techniques at the expense of foundational and theoretical skills as well as broader liberal-arts studies that are crucial for long-term career success and graduate work. The implications of these conflicting goals include allocation of finite resources (time, courses in the curriculum), unwillingness of students to invest in the mathematics that they see as irrelevant to their immediate career goals, and reluctance of faculty to have their courses be driven by a continually evolving marketplace of tools and APIs. A balanced curriculum benefits all stakeholders: students, employers, and faculty.

Would a data-driven approach help faculty design curricula that effectively balance these multiple goals? For example, if we ask graduates of computer science programs to reflect on the impact of their undergraduate education, explicitly focusing on short- and long-term impact, will there be enough meaningful data to significantly inform curricular design? A recent survey of industry professionals undertaken by the ACM/IEEE-CS/AAAI 2023 Computer Science Curricular Task Force (CS2023)[a] points the way. This column presents one aspect of that survey—a focus on comparing short-term and long-term views—and calls for similar surveys of industry professionals to be conducted on an ongoing basis to refine our understanding of the role played by various elements of undergraduate computer science curricula in the success of graduates.

IMAGERY FROM SHUTTERSTOCK.COM

The Survey

Approximately every decade, ACM along with its partnering organizations assembles task forces of international experts from academia and industry to assess changing trends in computing and issue curricular guidelines for undergraduate programs in computer science and six allied disciplines. The CS2023 Task Force on Computer Science curricula[2] started by conducting a survey of industry professionals, who were asked to provide their current background and rate various curricular aspects based on their own educational experience. The survey was encoded in Qualtrics, distributed in 2022 to a proprietary list maintained by ACM of its mem-

a See https://csed.acm.org/

bers and tech-talk registrants, and was completed by 865 practitioners. We focus here on aspects of the survey related to teasing out the differences between short- and long-term views of industry professionals.

The survey respondents were first asked to select the category representing their number of years of professional experience: 1–2 years, 3–5 years, 6–10 years, 11–20 years, and more than 20 years. Next, to elicit differences between short-term (first-job readiness) and long-term (career preparation) views, respondents were asked to rate the importance of 16 knowledge areas within computer science identified in CS2013[1]: algorithms and complexity, architecture and organization, computational science, discrete structures, graphics and visualization, human-computer interaction, information assurance and security, information management, intelligent systems, networking and communication, operating systems, platform-based development, parallel and distributed computing, programming languages, software development fundamentals, and software engineering.

Next, respondents were asked to rate on a Likert scale (0 = not applicable, 1 = not important, 2 = somewhat important, 3 = very important) how important the following curricular components were for preparing them for their career: capstone course or senior-project; internship or co-op while in college; availability of computer science electives; liberal arts education (including courses on history, languages, literature, philosophy, creative arts, psychology, and others); sciences (physics, chemistry, biology, and others), scientific method, and scientific inquiry; and mathematics (discrete mathematics, calculus, probability and statistics, linear algebra, and other mathematics).

Short- and Long-Term Views

The survey data presents an opportunity to answer two questions—when professionals are explicitly asked to take a short-term view and then a long-term view, are any areas of computer science rated differently? Second, do more experienced professionals (who are presumably able to take the long view) rate items differently than less experienced professionals? The purpose of this column is not to proclaim the importance of certain areas of computer science based on professionals taking one or the other view, but rather to make the case that there is a difference in short-term needs versus longer-term career preparation, and in the future, surveys of professionals should explicitly elicit these differences to better shape computing curricula.

Of the 16 knowledge areas listed, the five greatest differences between short- and long-term importance were reported in architecture, parallel and distributed computing, security, artificial intelligence, and databases. For example, the average rating (on a Likert scale of 0 = Not important, and 3 = Very important) of architecture was 1.93 for the short-term, but 2.50 for the long-term, a difference of nearly 30%. Interestingly, all five areas were rated more important for the long term. In some sense, the first three—architecture, parallel and distributed computing, and security—comprise a broader "systems" viewpoint. We propose that while first-job readiness might demand software development skills, the long-term view might emphasize foundational skills for system design.

Did more experienced professionals (21+ years in the industry) rate knowledge areas differently than less experienced ones (1–2 years or 3–5 years)? The biggest differences were seen in human-computer interaction and security, both rated more important by experienced professionals. Similarly, those with less experience rated artificial intelligence and databases more important than did more experienced professionals.

Ratings of the importance of curricular components (listed earlier) yielded more interesting results. Overall, the five components that were rated most important (averaged over both long- and short-term) were: probability and statistics, science, discrete math, linear algebra, and calculus. Yet, differences were found in the short- and long-term views: capstones and internships were rated more important for the short-term whereas probability/statistics and liberal arts were rated more important for the long-term. More experienced professionals rated science and liberal arts more important than did their less-experienced counterparts, who rated internships and capstones as being more important.

In summary, those taking the long-term view and those most experienced appeared to value knowledge areas and curricular components often seen as less immediately relevant by computer science students (architecture, probability, linear algebra, science, and liberal arts). *To prepare students for the long-term, we recommend that computing departments find ways to persuasively share this perspective with them.*

Those taking the long-term view and those most experienced appeared to value knowledge areas and curricular components often seen as less immediately relevant by computer science students.

Where to Go from Here: Curricular Adaptation

Although a survey dataset can numerically identify features of interest with clarity, what is less obvious is how curricula can be adapted in response. How exactly should a curriculum include science (for the long-term) and what kinds of capstone projects are sufficient (for the short-term)? How should a balance be struck between the two? We suggest a few principles to consider:

- Since a baccalaureate education is meant to last a lifetime and serve a variety of career paths, one should be careful not to sacrifice any area rated highly for the long-term. This is where a periodically administered survey such as this one that includes a long-term view comes in useful.
- It is incumbent upon educators to, at the very least, sufficiently prepare students for graduate study in the discipline.
- A curriculum that prioritizes foundational skills must also include development of market-ready skills. An example might be a capstone course that includes many first-job skills, such as software tool proficiency, professional ethics, writing, and presentation.
- When it comes to the development of theoretical (including mathematical) knowledge, many students are either averse to it or enter college underprepared. Yet, it is difficult for them to acquire this knowledge in the workplace—it is best acquired during undergraduate study. One way to motivate students to learn theoretical knowledge might be to combine it with applications. So, courses should be reformulated to include computing applications that use theoretical knowledge. Incidentally, some areas of theoretical knowledge might be more useful in industry than others, as the ratings for probability, statistics, and linear algebra indicate.
- While striving to craft curricula that balance first-job readiness with long-term career preparation, institutions should also try to effectively utilize their local expertise and communicate the particular student profile of their institution to potential employers.
- For recruitment, employers might want to use a combination of theoretical (for the long-term) and practical (for the short-term) assessment, which would signal to students the importance of also developing foundational skills.

A Call to Action: Future Data Collection

In closing, we make the case for periodically administering a national survey of computing industry professionals focused on curricular feedback. (A similar survey of faculty on the goals of computer science education was recently reported elsewhere.[3]) A rich and growing dataset from such surveys will help educators balance long-term value with first-job preparedness. More importantly, repeated surveys have the potential to dive deeper into rapidly changing areas such as artificial intelligence and data science, and elicit in sufficient detail, particular subtopics of consequence that may have arisen since prior surveys. Multiple rounds of surveys over multiple years also have a better chance of reaching a variety of computing practitioners. Finally, surveys, their analysis, and subsequent curricular refinement will lead to a better mutual understanding between employers and academia and reduce the gap in employers' expectations of computing graduates. C

References

1. ACM/IEEE-CS Joint Task Force on Computing Curricula. Computer Science Curricula 2013. *Technical Report.* ACM Press and IEEE Computer Society Press (2013); 10.1145/2534860
2. *Raj, R.K. and Kumar, A.N. Toward computer science curricular guidelines 2023 (CS2023). ACM Inroads 13*, 4 (Nov. 2022); 10.1145/3571092
3. *Valstar, S. et al. A quantitative study of faculty views on the goals of an undergraduate CS program and preparing students for industry. In Proceedings of the 2020 ACM Conf. on Intern. Computing Education Research (ICER '20).* ACM, NY (2020); 10.1145/3372782.3406277

Rahul Simha (simha@gwu.edu) is a professor in the Department of Computer Science at The George Washington University, Washington, D.C., USA.

Amruth Kumar (amruth@ramapo.edu) is a professor of computer science at the School of Theoretical and Applied Science, Ramapo College, Mahwah, NJ, USA.

Rajendra Raj (rkr@cs.rit.edu) is a professor in the Department of Computer Science at Rochester Institute of Technology, Rochester, NY, USA.

opinion

DOI:10.1145/3624638

Steven Fraser and Dennis Mancl

Opinion
Virtual and the Future of Conferences

Making conferences more accessible.

IMAGE FROM SHUTTERSTOCK.COM

DURING THE COVID-19 pandemic, conferences quickly transitioned to virtual. Based on our own virtual conference experiences and a survey we conducted in spring 2022, we see virtual conferences increased attendee access by reducing costs and travel time. However, the overall virtual conference experience during COVID was disappointing for some conferees. While technology and social norms will take time to evolve, we believe conference attendees should have a choice that includes virtual.

Access and Costs

Access is a challenge for prospective conference participants because many have fixed budgets and schedule constraints. Virtual conference attendance options will lower barriers to access. Our viewpoint is based on personal experience with recent COVID-era public conferences and internal corporate conferences,[5] where a hybrid blend of virtual and in-person conference options have been embraced for more than 20 years.

Companies organize multi-site internal virtual conferences to promote collaboration and knowledge transfer. Companies employ virtual technology to reduce their total costs by reducing or eliminating travel.

In contrast, for in-person public conferences, organizers focus on delivering a program rather than overall attendee costs. Even so, organizers should be sensitive to attendee costs (travel, hotels, and food) and environmental impacts (carbon costs). The carbon cost of air travel was identified as a factor in climate change by the Kyoto Protocol more than 20 years ago. Some companies, governments, and universities have limited employee travel as a part of their "green" initiatives or cost-containment strategies.[3] Additionally, virtual conferences enable contributions (keynotes, panels, workshops, tutorials) from experts who would not normally have the time or interest to attend an entire conference.

Even when organizers and attendees are faced with higher costs, there are many reasons for them to prefer in-person conferences. In-person conferences have intangible attractions for attendees that go beyond the serendipity of making new connections with other conferees. Early-career attendees enjoy the novelty of attending a conference and being inspired by experts in their field face-to-face. Conference regulars value informal in-person synch-ups with their colleagues. Many attendees learn from serendipitous hallway discussions.

The impetus of COVID-19 changed the equation, and it triggered a wave of experimentation and improvisation.

Some conference organizers reported hybrid sessions improved access and increased conference participation.[11,12] The adoption of virtual participation options can help reduce costs for attendees and open new markets for conference organizers. On the other hand, virtual options may add new expenses associated with IT infrastructure and video production staff.

On the negative side, many conference participants during the pandemic experienced disappointment. There were many logistical problems when conferences adopted virtual strategies at the last minute. Although virtual conferences and lower registration fees sparked increased registration, many attendees longed for in-person networking and social events. Screen fatigue, audio glitches, schedule snafus, and other unexpected experiences proved disappointing to attendees accustomed to in-person conferences. Additionally, virtual conference attendees missed opportunities for "bleisure" travel (combining business travel and leisure travel into one trip).[10]

Conferences with virtual options are not automatically successful. Both presenters and attendees face a learning curve for new virtual interaction models. Attendees must learn new hybrid interaction etiquette. For example, at a conference with a mix of in-person and virtual participants, in-person attendees might forget that when they ask a question, they need to use a microphone—so the virtual side of the audience can hear the question. Moderators, presenters, and audiences will need time for adoption of new virtual conference environments.

Across every area of science and technology, it is essential to encourage diversity and the engagement of underserved parts of the community. The ACM Code of Ethics[1] calls on us to be an inclusive community: "Computing professionals should foster fair participation of all people, including those of underrepresented groups." Some members of the global computing community face economic or political barriers to conference access. For students, early-career professionals, and people who are physically challenged, access to in-person conferences may not be easy. Of course, even a registered conference attendee might miss an in-person conference due to illness or a work or family schedule conflict. Another barrier to conference participation is long-term family responsibilities, such as childcare or elder care. Family-care responsibilities disproportionately affect women, at a time when our community wants to encourage equal access. Virtual and hybrid conferences lower the access obstacles and help serve a global community.

There is a growing body of literature on the challenges and best practices for virtual and hybrid conferences.

Challenges

There is a growing body of literature on the challenges and best practices for virtual and hybrid conferences.[2,4,9] This literature reveals many strategies, but one size does not fit all. Each event is different, and each sponsor must analyze the needs of that conference's stakeholders, so the program is accessible to all prospective attendees.

Virtual conference interaction is challenged by "invisible audiences." Without tacit audience feedback through facial expressions or body language, presenters may unknowingly "lose" their audiences. Virtual meetings must evolve to better support interactive dialogues such as those we experience in person.

While virtual conferencing platforms can deliver conference content to a global audience, session moderators, and to a lesser extent audiences, must adopt practices to foster increased engagement. For example, hybrid conferences must balance questions from in-person and virtual attendees, where it is too easy for in-person attendees to miss or (deliberately ignore) input from virtual meeting participants. A session moderator must give both local and virtual questioners an equal opportunity to be heard.

For informal virtual conference interactions, new mechanisms will undoubtedly emerge as social norms and technologies evolve. Session recordings will help mitigate time zone issues, or conferences might be reorganized with shorter sessions spread over more days.

Some prospective conference attendees will continue to be motivated by bleisure travel opportunities. These attendees will resist the transition to virtual.

Academic conference organizers face a special challenge if virtual conferences become more common. Pre-COVID, acceptance of a research conference paper required the presence of at least one author. However, if virtual presence is sufficient, as proposed by Jason Harline in a July 2023 BLOG@CACM item,[8] then it may be more difficult to attract in-person audiences given travel and environmental costs.

Community Survey

Motivated by curiosity, the authors ran a community survey in spring 2022 with the goal of learning more about in-person and virtual conference attendee attitudes.[6,7] Responses covered a range of geographies and included input from industry practitioners and university researchers.

The survey results indicated more than 60% of respondents preferred either virtual or hybrid conference options. Among the 331 survey respondents, the split was 54% preferred hybrid, 36% in-person, and 9% virtual. Respondents shared a wide range of opinions. Some respondents were in favor of hybrid meetings so they could be flexible in their choice of how to attend, other respondents could not wait for in-person meetings to resume.

Respondents also ranked potential conference obstacles. For virtual conferences, two-thirds of respondents identified "ineffective support for casual discussions" as a major obstacle. Almost half (49%) reported that "fatigue due to long virtual meetings" was a major obstacle. For in-person conferences, the most important obstacles were (in spring 2022) "ongoing pandemic related health risks" (45% rated as a major obstacle) and "registration and travel costs" (40% rated as a major obstacle).

Increased adoption of virtual conference platforms will help reduce costs, increase access, and extend global reach.

Survey respondents indicated their near-term plans for attending conferences. Many respondents were planning to attend more conferences per year—based on their experiences with virtual and hybrid conferences in 2020 and 2021. The average number of conferences attended per respondent was 2.0 in 2021, and the respondents planned to attend an average of 3.5 conferences in 2022. The authors plan to run a follow-up survey in spring 2024.

Trends and Call to Action

Conferences will continue to be an important catalyst for attendees to share ideas and make connections. Increased adoption of virtual conference platforms will help reduce costs, increase access, and extend global reach. However, as with virtualization of retail commerce, health care, education, and government services, change will take time.

A conference platform should be more than a meeting portal. To recreate the dynamics of an in-person conference, a virtual conference platform needs to support the serendipity of personal and group interactions and enable complex presentations and discussions. Recorded presentations not only enable asynchronous conference viewing but also serve as a post-conference record—much in the same way journals and conference proceedings served in the past.

We foresee these trends for future conferences:

- More virtual options;
- Fewer barriers to attendance;
- Increased global participation;
- Experimentation with session design and attendee engagement; and
- Evolving attendee expectations

We feel that without some platform standardization, different conferencing applications will create unnecessary challenges for both organizers and participants. ACM and other conference sponsors can help by encouraging platform vendors to enhance the usability and interoperability of their meeting platforms.

Conferences must become more accessible to a prospective global audience by reducing financial and environmental costs. Our survey suggests there is a community desire for a blend of virtual and in-person conferences.

However, while not every conference must include virtual options, conference sponsors should not ignore the economic, technological, cultural, political, and climate forces at play. Our connected culture has changed our work environment, and while a wave of nostalgia may motivate a short-term return to in-person conferences, we call upon the community to take greater advantage of the accessibility and economics of virtual conferencing. ◼

References

1. *ACM Code of Ethics and Professional Conduct*. (2018); https://www.acm.org/code-of-ethics
2. *ACM Presidential Task Force on What Conferences Can Do to Replace Face to Face Meetings*. (2020); https://www.acm.org/virtual-conferences
3. Eriksson, E. et al. On the necessity of flying and of not flying: Exploring how computer scientists reason about academic travel. In *Proceedings of the 7th Intern. Conf. on ICT for Sustainability (ICT4S2020)*, (2020); doi: 10.1145/3401335.3401582
4. Ferlin, S. et al. *Reviewing Best Practices in Online Conferencing*. (2021); https://arxiv.org/abs/2212.11619
5. Fraser, S. Strategies for the future of work: Accelerating innovation through tech transfer. *Agile Alliance Experience Report*. (2021); https://bit.ly/47LxEjD.
6. Fraser, S. and Mancl, D. *Report on the Future of Conferences*. (2023); https://arxiv.org/abs/2301.03544
7. Fraser, S. and Mancl, D. *The Future of Conferences Research Survey*. (2022); https://manclswx.com/survey2022.html
8. Harline, J. The dissemination game: Incentives of in-person vs. virtual participation. *BLOG@CACM*, (July 26, 2023); https://bit.ly/3Rw3dbw.
9. Lopes, C. *The Future of Conferences, Strange Loop 2021 Conference*. (2021); https://www.youtube.com/watch?v=LkJNA88R_5w
10. Morgan, B. What is bleisure travel, and how is it transforming the hospitality industry? *Forbes*, (Jun. 27, 2022); https://bit.ly/41lFzSr
11. Potanin, A. *Hybrid SPLASH 2021 Retrospective* (2022); https://blog.sigplan.org/2022/08/25/hybrid-splash-2021-retrospective/
12. Skiles, M. et al. Conference demographics and footprint changed by virtual platforms. *Nat. Sustain 5* (2022). doi: 10.1038/s41893-021-00823-2

Steven Fraser (sdfraser@acm.org) is Director, Innovation, Partnerships, and University-Company Collaborations at Innoxec, Santa Clara, CA, USA.

Dennis Mancl (dmancl@acm.org) is a software expert at MSWX Software Experts, Bridgewater, NJ, USA.

DOI:10.1145/3624726

Alfred Z. Spector

Opinion

Gaining Benefit from Artificial Intelligence and Data Science: A Three-Part Framework

Why ethics is not enough.

RESEARCH AND PRODUCT teams in artificial intelligence (AI) and data science need to think more deeply about how their work maximizes benefits and mitigates harms—particularly as their efforts are applied to increasingly challenging problems with diverse, global impacts.[7] Achieving good results may entail making complex trade-offs, especially when initial goals are difficult to specify and disputed and when a project's initial objectives have complex, long-term side effects.

In this column, I argue we cannot make these trade-offs by an appeal to ethics alone. Instead, we must:

- Begin with a deep understanding of the field-specific challenges we face;
- Strongly commit to integrity; and
- Leverage understanding from ethicists, but also *engage with the broader liberal arts communities* (economics, political science, history, literature, and so forth) to better envision the impact of our decisions and gain the needed breadth and depth their expertise can provide. As Brodhead writes, our "dynamism of a culture of innovation has come in large part from our distinctive liberal arts tradition, in which students are exposed to many different forms of knowledge and analysis, laying down a mental reservoir that can be drawn on in ever-changing ways to deal with the unforeseeable new challenges."[3]

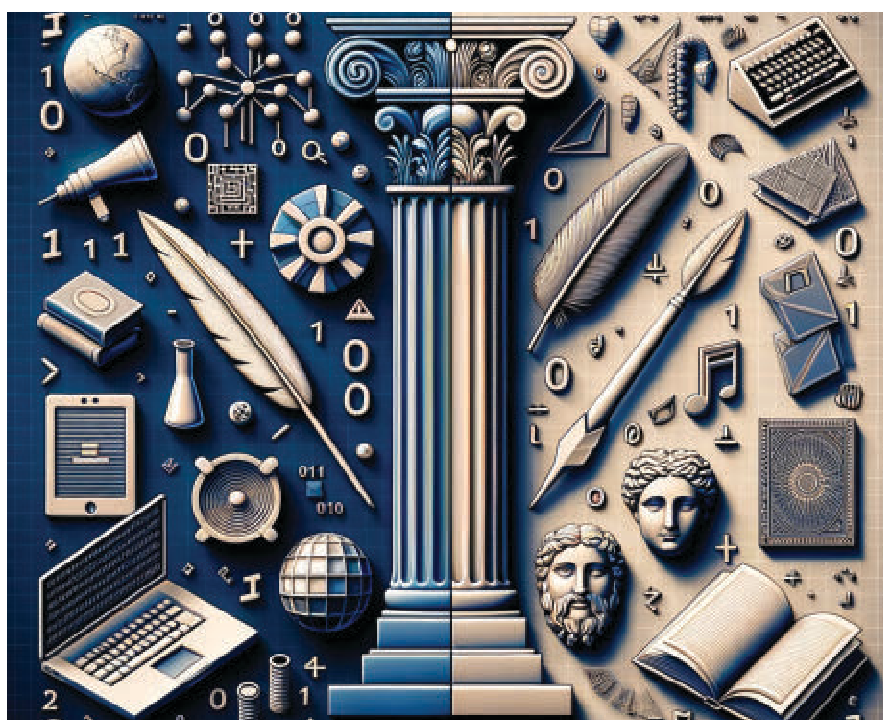


IMAGE CREATED BY ANDRIJ BORYS ASSOCIATES WITH THE ASSISTANCE OF SHUTTERSTOCK.AI

Only by the application of these three items can we apply our ever-more-powerful technologies to achieve the best results.

A Journey

I have had the opportunity to read and reflect on this topic while co-authoring *Data Science in Context: Foundations, Challenges, Opportunities*,[14,15] where my co-authors and I enumerate, categorize, and describe the complex issues that arise when applying data science. These "gotchas" led us to create an "Analysis Rubric"' containing top-level elements encompassing technical and societal considerations, as shown in the table in this column.

The rubric is intended to help scientists, engineers, and product managers as well as social scientists and policymakers ensure they have considered the breadth of issues needed to apply data science beneficially. The rubric also identifies and motivates many research challenges, such as causal in-

Field-specific challenges in data science.[15]

Implementation-Oriented Elements	Requirements-Oriented Elements
Tractable Data: Feasibility of gathering and holding data of sufficient integrity, size, quality, and manageability.	**Understandability:** Capability to explain results, demonstrate causality, or allow results to be reproduced.
Technical Approach: Availability of techniques of analysis, models, and visualizations that can meet goals.	**Clear Objectives:** Existence of well-specified objectives that align with what we truly want to happen.
Dependability: Ability to meet privacy, security, abuse-resistance, and resilience needs.	**Toleration of Failures:** Acceptability of direct and unintended consequences of objectives or poor results.
Legal and Societal Issues: Ability to comply with increasingly complex legal and regulatory issues, achieve beneficial societal impacts, and mitigate environmental and other harms.	

ference, detecting abuse or fake data, and providing interpretable machine learning. While this list focuses on data science, many of the topics are adaptable to AI, computing research more broadly, engineering, and product development.

Contemporary embedded or standalone technology classes in many universities include case studies relating to these rubric topics. For example, The University of Toronto's Embedded Ethics Education Initiative has a module that discusses the societal benefits and counterbalancing privacy risks of COVID-19 contact tracing, and another that presents the complexity of setting the right objectives in recommendation systems.[5] One of MIT's modules discusses the balance of privacy versus data quality relating to the use of differential privacy in the census; another explores the complex goals and impacts of algorithmic approaches to defining voting district boundaries.[11] A Stanford case study explores the potential safety benefits of self-driving vehicles versus their differential impacts on sub-populations.[9]

Unfortunately, the identification of challenges and trade-offs does not specify how to balance competing goals. This complexity becomes manifest as one considers the long-term, widely dispersed consequences of decisions.

In considering this problem, my co-authors and I realized that any discussion of trade-offs must start with integrity, which is the foundation (in computer science terminology, the "secure kernel") for the proper conduct of all science and engineering endeavors. In data science and AI, this has become ever more apparent as we are confronted with visible risks arising from the misuse and misrepresentation of data and extending to computing's broader impacts. As professionals, we must disclose the limits of our art, practice lawful behavior, always tell the truth, and not misrepresent our conclusions or capabilities.

With a foundation in integrity, we turned to the Belmont Principles for applied ethics in biomedical and behavioral research as a useful ethical framework.[16] The three Belmont principles can be applied more broadly and particularly to applications of data science.

- *Respect for persons* supports an individual's freedom to act autonomously and be free from undue manipulation. In medical research, it often surfaces as "informed consent," while in applications of data science, it serves as the rationale for clear disclosure, "opt-in," and similar policies.
- *Beneficence* refers to the need to maximize benefits and balance them against risks. Negative consequences may be unanticipated or emergent phenomena, and they require continual monitoring and vigilance. In healthcare, beneficence is often associated with "first, do no harm," but this is overly simplistic as almost every treatment has some risk.
- *Justice* requires attention to the distribution of risk and benefits across a population. In human subject research, Belmont suggests that there should be particular benefits to the populations who bear risk from the research. In applications of data science, considerations of justice extend to broader considerations of fairness across larger populations.

Unfortunately, the identification of challenges and trade-offs does not specify how to balance competing interests.

Certainly, there are other important and relevant philosophical principles beyond Belmont. Here are a few with particular relevance today:

- Aristotle argues that virtue serves the good of our community and, at the same time, is necessary for our own happiness or completion.[1] His writing presages the need for considering the societal impact of technologies that individuals choose to adopt. Shannon Vallor's recent book *Technology and the Virtues* focuses on the need to apply virtue ethics to AI-related issues.[17]
- John Stuart Mill espouses the Harm Principle, which states "the only purpose for which power can be rightfully exercised over any member of a civilized community, against his will, is to prevent harm to others," but he also advocates "exertion to promote the good of others."[10] His two statements help us understand differing views on the regulation of technology, the definition of fairness objectives, and many other topics we read about daily.
- The principles leading up to war (jus ad bellum) and conducting war (jus in bello) may inform our views on the application of data science and AI in military domains,[12] for instance in the context of Russia's invasion of Ukraine. Ethical principles are different in such extraordinary and harmful circumstances.

Experts would add more philosophers and philosophical frameworks to help guide our thinking.

With this column's first emphasis on technical elements (for example, dependability) and its second advocacy for honesty and ethics (for example, Belmont), readers might surmise we have a complete framework for educating and guiding ourselves toward the effective and constructive uses of our technologies. Upon reflection, I have concluded these two components are both necessary but not sufficient. This is because there are other frameworks and models of thought (not usually labeled as ethics) that are equally critical to the right and proper uses of technology. While ethics tends to concentrate on what world we want, there must also be focus on the pragmatics of how societies organize themselves to achieve practical progress toward their goals. There are many situations where certain ethical objectives may not be directly achievable due to the strains they put on a political system, the economic inefficiencies they cause, or other long-term consequences.

We thus need not just an understanding of ethics, but also of other disciplines that enable us to reach the best possible conclusions:

▸ **Economics** focuses on the study of systems that foster logical and effective use of resources to meet private and social goals, including such objectives as economic freedom, fairness in reward and the balance of wealth, economic security, growth and efficiency, price stability, and employment. Economics treats topics such as the balance between equity and efficiency, economic freedom and social welfare, and incentives to promote (or lessen) desired (or harmful) economic results or behaviors. In a market economy, profit and its associated surplus catalyze innovation and may conflict with ethical principles that might espouse redistribution to foster fairness.

▸ **Political science** focuses on the study of societal governance, often with the objective of developing more effective political systems. Based on consideration of political theory, comparisons of different political systems, and international relations, the field illuminates the impact of technology on societies and their governments and also informs us as to the ability to effectively apply regulation or legal mandates. Topics such as political processes, fairness, liberty, justice, stability, international order, and more arise.

▸ **History** focuses on explaining, evaluating, and otherwise analyzing historical events that have transformed societies, often to inform us of the potential impact of decisions. Churchill, channeling Santayana, is widely quoted as saying, "Those who fail to learn from history are condemned to repeat it."[13]

We must turn to more fields of study, whether other social sciences or the arts, if we are to truly leverage society's accumulated wisdom.

▸ **Literature** is the study of texts with the goal of increasing understanding of the human condition—the world around us: as it was, is, and may be.

Examples abound where the contemporary decisions on the application of technology require broad considerations of not just ethics, but economics and political science. Targeted advertising is a good example:

▸ Examples of ethical concerns arise from all three Belmont Principles: *respect-for-persons* addresses issues relating to opt-in, *beneficence* requires mitigating harmful manipulation, and *justice* considers the potential for societally imbalanced benefits and harms.

▸ Economic concerns arise from the role of advertising in catalyzing new and more efficient markets, creative endeavors, innovative products, and much of the Web ecosystem while balancing potential negatives, including concentration of power and wealth.

▸ Political considerations include balancing the likelihood that governmental entities can add value without undue harm to societal innovation, prosperity, or their own legitimacy.

▸ From a historical perspective, there is much to learn about unintended consequences of seemingly good decisions—for example, where regulation has worked reasonably well (for example, medical advertising), and where it has led to poor outcomes (for example, incessant cookie warnings).

▸ Fiction may portray dystopian worlds that warn us of harmful consequences or delight us with the prospect of changes, which we might otherwise narrowly reject.

This list of relevant disciplines is limited and may still not supply sufficient perspective. We must turn to more fields of study, whether other social sciences or the arts, if we are to truly leverage society's accumulated wisdom. The recent book, *Framers*, which broadly discusses the models underlying human decisions, argues that great breadth is needed for truly insightful decisions.[8] More directly, Connolly's provocatively titled piece, "Why Computing Belongs within the Social Sciences," argues compellingly that we should "Embrace other disciplines' insight."[6]

A Three-Part Framework

This discussion motivates the main thrust of this column: A three-part framework that we, as practitioners and educators, can apply to guide computational and data-centric technologies to achieving the most beneficial results.

1. We, as computer scientists, data scientists, experts in machine learning, and other subjects, must teach and attend meticulously to our own *field-specific challenges*. Just as structural engineers must teach and apply the consequences of metal fatigue or oxidation, our field requires in-depth research, understanding, and careful application of the topics contributing to dependability, understandability, fairness, sustainability, and others appearing in the accompanying table.

2. We must also practice, teach, and demand *integrity*. We must be scrupulously careful to disclose the limits of our capabilities, to practice lawful behavior, and to always tell the truth. We must act to prevent the growing amounts of societal dishonesty from entering our field and our practice. I explicitly call out integrity as this second item because we can be individually responsible for it and make it part of all our courses, it is often clearer than other topics in ethics, and integrity is the foundation of trust, reproducibility, and reasoning.

3. We and our students must have increased grounding in the *liberal*

We should have the humility to recognize we cannot become experts in all relevant disciplines, so we must work collaboratively with others.

arts. As our research and work products are applied to ever more complex problems, they will require trade-offs that rely not only on ethical considerations, but equally on economics, political science, history, other social sciences, and the humanities. When we educate computer science or data science students, we should ensure they take a sufficient number of courses in other disciplines, and we should explicitly advise them on their non-technical curricular choices. This suggestion is broadly consistent with Connolly's view[6] but advocates leveraging a students' core curricula or distribution requirements more than replacing technical courses. We should have the humility to recognize we cannot become experts in all relevant disciplines, so we must work collaboratively with others. Just possibly, as programming becomes increasingly automated,[2] the distinguishing characteristic of our field's most successful contributors may tilt even more toward those having this breadth of education.

We must be broadly educated if we are to guide the growth of computing, data science, and AI and its application to ever more difficult, even wicked,[4] problems. As technical experts, we certainly need to both lead and educate our students on our field-specific challenges and lead in the education and practice of integrity. However, we and our students also must have broad grounding in the non-technical topics that influence whether our research or products can achieve the positive results we desire. These topics include ethics, but also the other enablers of our societies including economics, political science, history, and more. As educators, we must guide our students to an important collection of non-technical courses. We also must increasingly collaborate with professionals in other fields, as we cannot be experts in everything. **C**

References

1. Aristotle. *Aristotle's Nicomachean Ethics*. University of Chicago Press (2021).
2. Bird, C. Taking flight with Copilot: Early insights and opportunities of AI-powered pair-programming tools. *Queueing Syst. 20*, 6 (Jan. 2023).
3. Brodhead, R.H. On the tenth anniversary of the heart of the matter. *Bulletin of the American Academy of Arts and Sciences 76*, 6 (2023).
4. Churchman, C. Free for all: Wicked problems. *Manage. Sci.* (1967).
5. Computer Science, University of Toronto and Schwartz Reisman Institute for Technology and Society. *Embedded Education Ethics Initiative (E3I)*. (2023); https://www.cs.toronto.edu/embedded-ethics/
6. Connolly, R. Why computing belongs within the social sciences. *Commun. ACM 63*, 8 (Aug. 2020).
7. Crawford, K. *The Atlas of AI: Power, Politics, and the Planetary Costs of Artificial Intelligence*. Yale University Press. (2021).
8. Cukier, K. et al. *Framers: Human Advantage in an Age of Technology and Turmoil*. Penguin (2022).
9. Cohen, H. et al. *Stanford Computer Ethics Case Studies and Interviews*. (2023); https://bit.ly/3TLvAnM
10. Mill, J.S. *On Liberty by John Stuart Mill*. Longmans, Green, and Company. (1867).
11. MIT Schwarzman College of Computing. *Case Studies in Social and Ethical Responsibilities of Computing*. (2023); https://bit.ly/3HaEd3D
12. Moseley, A. A just war theory. *Internet Encyclopedia of Philosophy*. (2023); https://iep.utm.edu/justwar/
13. Santayana, G. *Introduction, and Reason in Common Sense*. C. Scribner's Sons (1905).
14. Spector, A.Z. et al. More than just algorithms: A discussion with Alfred Spector, Peter Norvig, Chris Wiggins, Jeannette Wing, Ben Fried, and Michael Tingley. *Commun. ACM 66*, 8 (Aug. 2023).
15. Spector, A.Z. et al. *Data Science in Context: Foundations, Challenges, Opportunities*. Cambridge University Press. (2022).
16. United States National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research. *The Belmont Report: Ethical Principles and Guidelines for the Protection of Human Subjects of Research*. (1978).
17. Vallor, S. *Technology and the Virtues: A Philosophical Guide to a Future Worth Wanting*. Oxford University Press (2016).

Alfred Z. Spector (azs@azs-services.com) is a visiting scholar at Massachusetts Institute of Technology, Cambridge, MA, USA.

The author appreciates the valuable comments of Lauren Cowles, Peter Hansen, Mehran Sahami, and the anonymous reviewers, and thanks Reynold Spector for his continuing inspiration.

Watch the author discuss this work in the exclusive *Communications* video. https://cacm.acm.org/videos/gaining-benefit

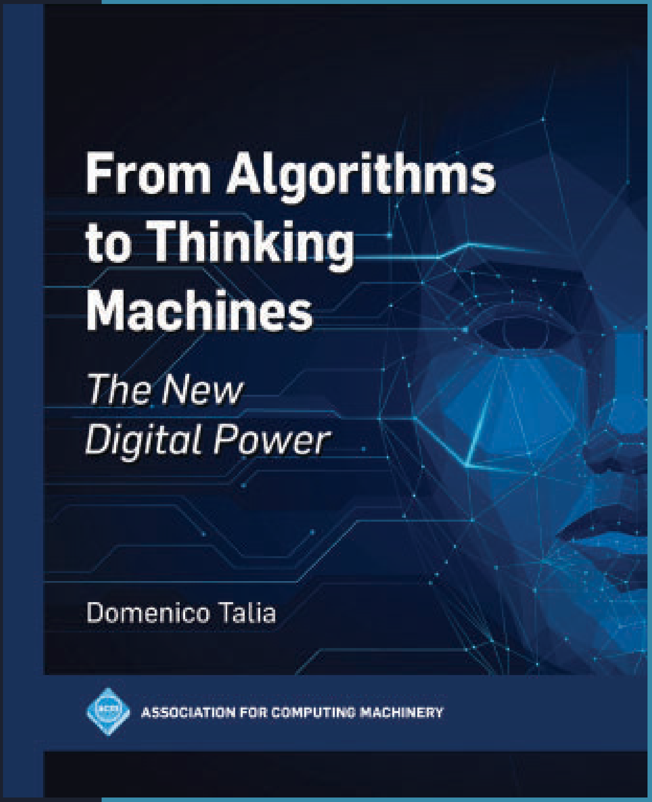
ACM BOOKS

Collection III

This book introduces and provides an analysis of the basic concepts of algorithms, data, and computation and discusses the role of algorithms in ruling and shaping our world. It provides a clear understanding of the power and impact on humanity of the pervasive use of algorithms.

From Algorithms to Thinking Machines combines a layman's approach with a well-founded scientific description to discuss both principles and applications of algorithms, Big Data, and machine intelligence. The book provides a clear and deep description of algorithms, software systems, data-driven applications, machine learning, and data science concepts, as well as the evolution and impact of artificial intelligence.

After introducing computing concepts, the book examines the relationships between algorithms and human work, discussing how jobs are being affected and how computers and software programs are influencing human life and the labor sphere. Topics such as value alignment, collective intelligence, Big Data impact, automatic decision methods, social control, and political uses of algorithms are illustrated and discussed at length without excessive technical detail. Issues related to how corporations, governments, and autocratic regimes are exploiting algorithms and machine intelligence methods to influence people, laws, and markets are extensively addressed. Ethics principles in software programming and human value insertion into artificial intelligence algorithms are also discussed.



From Algorithms to Thinking Machines

The New Digital Power

Domenico Talia

ISBN: 979-8-4007-0855-8
DOI: 10.1145/3603178

http://books.acm.org

DOI:10.1145/3630746

Article development led by acmqueue
queue.acm.org

Security must be a business enabler, not a hinderer.

BY PHIL VACHON

Security Mismatch

A "DING!" SOUNDS from your computer: You've got mail. Like Pavlov's well-trained dog, you open your email client. Something dreadful lurks at the top of your inbox: updates to a security review ticket. As you review the findings, you despair. Your team is already late on delivering features to paying customers, and you don't have the time or resources to fix everything. When the security team gets involved, the goalposts seem to move. It sometimes feels like every reviewer finds some theoretical issue they have just read a paper about and, with that, some new reason to delay your launch date. You schedule an impromptu meeting with your product management team to review the plan.

Maybe they will be flexible …

Anyone who has lived the product delivery life cycle at any mid- to large-size corporation has experienced some version of this moment—a security team gives a list of problems at the last minute, and the product team, already running late, weighs the risk of ignoring product security's guidance. Conversely, the security reviewers are brought in late in the game, give the product team a list of things to do before launch that isn't prioritized, and are unclear about how bad these issues are.

Too often, overworked product security reviewers must chuck preliminary findings over the transom, leaving it to the product team to sort out. The unconstructive feedback loop continues.

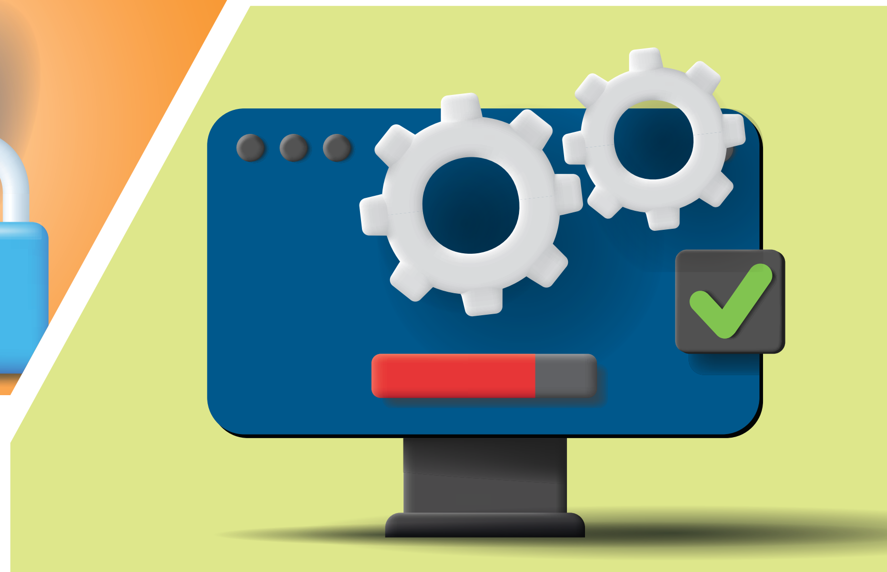
Worse, security teams eye product delivery teams warily, as if they are guilty of a mortal sin, their apparent ignorance of security best practices leaving them beyond salvation. On the other side, product delivery teams view security teams as a bunch of highly paid cowboys who cook up implausible and unrealistic risk scenarios. Product teams crave clarity about which high-priority risks to address—after all, security is exciting. It's not uncommon to see security teams fail to capitalize on this excitement, though, turning these interactions into something product teams dread. Isn't there a way to bridge the gap?

The product security reviewer breathes a sigh of relief. Another ticket done, for now, until it bounces back into the pending reviews queue. There is increasing pressure to get through more of these tickets. There isn't great guidance for the security team, either—what risks is the business, which is driving the product, willing to accept? What are even the most important products to examine for problems?

A "ding!" sounds from the security reviewer's computer. Sighing, she opens her email client. Something lurking at the top of her inbox: an all-hands update from the chief information security officer, thanking the team for all their hard work.

The "thank you" buries the lede: It is followed by a reminder the company is in a hiring freeze. But the company keeps shipping new products, so how can her team keep up with the load? Resigned, she looks at the top of the security review ticket queue: Next on deck is a ticket she has reviewed before. Maybe she will find something new …

Software is inherently complex. The economic pressures in developing software systems exacerbates this fact. The boom of frameworks, service-oriented architectures, pervasive code reuse, and



IMAGERY BY MARYNOVA

other complexity management strategies in software engineering helps reduce the "from scratch" costs.

Some might argue these strategies can take a complex design and turn it into an even more complicated problem, but a corollary of Hyrum's Law[a] is that software systems are often inadequately specified. It's in these ambiguities of specification that opportunities arise for software developers to foster their own interpretations of what is part of a public interface or a feature of a larger system, and what is not. Security teams live at this level and thrive on these ambiguities. All a product team wants to do is get something out that works for well-behaved users.

MIT's Nancy Leveson, professor of aeronautics and astronautics, reminds us often that reliability, resiliency, and security are emergent properties of a well-defined system.[b] A product is a client-facing part of a system, one that rests upon many thousands of lines of infrastructure code, all the way down to the metal and silicon of the underlying hardware. Each of these layers has its own management, reliability, and security challenges, and the development teams that own things higher up the stack are abstracted away from the details found on the lower layers. Often, security features are bolted on as an afterthought, or worse, added years after the original developers have left the company.

You can see where the mismatch occurs: Security teams exist in a very different space. They dive through abstractions, building a deep understanding of how specific components interact. They often lose track of the details that bind the layers together, however, or the fact that product teams don't even have insight into the structure of these layers. Or they may make optimistic assumptions about the complexity of the "simple" changes they ask for—your "simple interface" change could blow up into an exponential number of places.

You huddle in a conference room with your team. Your product manager, who is based at corporate headquarters on the other side of the continent, is on Zoom. The security reviewer was not clear about which problems were the worst, so you must make your best guess at priorities.

The product manager looks disinterested as you rattle off the list of findings and proposed resolutions. She interrupts and asks, "Look, is the security team stopping us from shipping?" You ponder this for a moment, filled with dread knowing what is coming next.

She sighs. "Well, ship it. We will deal with the fallout. The board cares more about revenue than what the security people say." She signs off the Zoom call. She evidently has more important things to worry about …

Because of these differences in world view, security teams tend to find themselves at odds with product teams. Even with the best of intentions, they find their guidance being ignored.

Security teams start to fall back on fearmongering to justify why their work is important to a business. This creates more friction, and an us-versus-them mindset finds fertile ground in these environments. The security team that says no to everything is a common trope in the modern corporate environment, but this response isn't given out of malice. Sometimes it's just a matter of being overwhelmed and not having good ways to answer, "Yes, but …"

Security teams must tactfully remind their partners that attacks on corporate infrastructure are lucrative for the bad guys, especially in an era of ransomware and data extortion. The sophistication of these types of attacks is only increasing, too—but criminals are not using any novel techniques to get their initial toeholds. It's amazing how weak authentication to important services, unpatched or sensitive systems being exposed to the public Internet, and social engineering are still at the root of many high-profile attacks. (Someone clever might ask, "Why don't you include 'disgruntled insiders' or '0-days' on the list?" The former is a unique business challenge to address, but it is a risk. The latter is improbable unless you are really in the wrong place and are being targeted by the right people.

One thing's for certain: Information security teams that say no must change. (Note that I never use *cybersecurity* to describe what we do; I'll tie myself into knots to avoid using the word *cyber* if I can. This is just a preference.) Hiding behind a moat makes repelling attacks easy, but bridges allow you to replenish supplies and foster relationships with customers' castles. Remember, a security team's role is to empower their business to move forward with confidence, not to hinder progress. C

a https://www.hyrumslaw.com/
b http://sunnyday.mit.edu/safety-3.pdf

Phil Vachon leads the information security program at Bloomberg's CTO's office in New York City, NY, USA. Previously, he co-founded and was CTO of a startup that built a high-speed packet capture and analytics platform and worked on spaceborne synthetic aperture radar data processing and applications.

DOI:10.1145/3630745

Article development led by acmqueue
queue.acm.org

Personal, team, and organizational effectiveness can be improved with a little preparation.

BY THOMAS A. LIMONCELLI

Knowing What You Need to Know

BLOCKERS CAN TAKE a tiny task and stretch it over days or weeks. Taking a moment at the beginning of a project to look for and prevent possible blockers can improve productivity. These examples of personal, team, and organizational levels show how gathering the right information and performing preflight checks can save hours of wasted time later.

Example 1: Personal Productivity.

Two IT workers—Andrew and Bertie (not their real names)—were assigned the same task. While this task should have taken about an hour of hands-on keyboard work, it took Andrew four days.

Andrew's story. Andrew began the task one sunny Monday morning. Work went well until he hit a speed bump and needed to ask the requester (who we will call Roger) a question. Andrew tried to find him on the company chat system, only to learn Roger was out of the office. Andrew sent an email instead.

Roger replied on Tuesday, which was a very busy day for Andrew, who didn't see the reply until the afternoon. By then, he was still busy with something else and didn't get to work on Roger's request until Wednesday.

On Wednesday, Andrew made some progress but had more questions. Roger's request referred to "my database," and Andrew, smartly, decided to contact Roger one more time to ask the exact name of the database rather than guess.

Andrew did not have the necessary access rights on that database, and it took a while to find a coworker who could grant him the required access.

Andrew finally completed the task on Thursday—about one hour of work that took four days to complete.

Bertie's story. Bertie also began the task on Monday morning. She reviewed the request and quickly realized there was a lot of missing information. She called the requester's cell phone and interviewed him until she had everything she would need to do the task without interruption. While still on the phone, she verified she had the required database access. She didn't, but the person was able to grant her permission while on the phone. In a few minutes, she had everything needed to get started. She completed the request shortly afterward, in about two hours total.

Same task. Very different levels of productivity.

Analysis. Bertie understood the value of gathering everything required to do a task up front. Andrew was trying to get the information he needed "on demand" or as he got to

each step. To Andrew, it felt more productive to dive in and begin right away. Sadly, Andrew was wrong.

As a general principle, people are more productive when they have all the information needed to do a task up front.

With this in mind, what improvements could be made?

On a personal level, at the start of a task you can pause to think through the entire process. Be pessimistic and ask yourself, "What could go wrong?" Verify that needed access is available. Verify the information required to do the task is complete.

On a team level, Bertie could teach the rest of the team what information to collect for that kind of task.

This can be done informally by casually reminding the team at their weekly staff meeting what information needs to be gathered before starting such a task.

It could also be done formally. If there is a process document, it can be updated to include a "before you begin" section. This would list information to collect, as well as other preflight checks such as verifying access rights, listing which software tools should be installed ahead of time, and so on.

By recording such lists in a way that others can use and reuse, Bertie boosts the productivity of anyone who does the same task in the future. Not only is she more productive, she makes everyone more productive.

Am I allowed to do this? I suspect some professionals reading this might work in environments where they do not have the latitude to create such preflight checks for peers. In these environments, process improvements are a management responsibility, and it would be presumptuous for a subordinate to try to improve anything. This may seem counterproductive—because it is. Process improvement is everyone's job, and when managers create roadblocks that prevent the people doing the work from making improvements, they do a disservice to the company.

Such restrictions are counterproductive, yet I suspect they are more common than you would expect. In such environments, the creation of preflight checklists like Bertie's might take months of bureaucratic wrangling and approvals. It would be faster to do it unofficially, by just passing notes around or setting up a shadow documentation repository. Doing this unofficially, however, might upset the political apple cart or agitate an overly zealous manager.

My advice is to do it anyway. A good manager will reward such initiative and a bad manager will simply reveal yet another reason why you should be sending out your resume.

One of the reasons I like keeping procedure documents under source-code control is that improvements can be proposed, reviewed, approved, and merged just as we do with code changes. For example, Git's pull request (PR) process, normally used for code changes, works well for documents and affords the same review, tracking, and accountability.

Readers who work in highly regulated environments might think they cannot just change a process on a whim, and it is easier to live with a broken process than go through the steps to fix it. But no regulatory body requires you to keep buggy processes. In fact, they usually have requirements that broken processes be fixed in a timely manner. Git PR provides a way to make a change with approvals, tracking, and accountability that enable you to comply with regulations more efficiently.

Example 2: Transactional Requests

Receiving a request by email is convenient for the requester, but it is prone to not collecting all the information needed to complete the request.

In the old days, there were paper forms that people filled out with a technology commonly known as a *pen* (kids, ask your grandparents). The form collected the pertinent information up front.

Some companies had dozens or hundreds of forms. When I worked at Bell Labs in the 1990s, the stockroom had a huge shelving unit with a compartment for each form. "A form for every occasion," we used to joke as we walked by it on the way to lunch.

There was even a form for ordering more forms! I always wondered what would happen if we ran out of that form. I suppose the company would have simply ground to a halt and ceased to exist.

Today, a typical helpdesk request is submitted via email or a Web form with a single huge text box. The requester is free to format the request in any way and include as much or as little detail as desired. This is particularly frustrating and anxiety-inducing for users. I've seen countless helpdesk tickets begin with an apology for not knowing how to ask what they're about to ask. How toxic is it that we create systems that, by design, make its users feel ashamed of themselves?

There is comfort in being told what to do by a form. It does not judge you for not knowing what options are available and instead, lovingly provides checkboxes and blanks that hold your hand through the process of making the request.

Modern helpdesk systems solve this by providing an electronic form facility so that the correct information is collected at the outset. Creating an electronic form for every occasion is difficult, however, and most IT teams I have seen use the form feature sparingly, often only for a few standardized requests such as asking for a new laptop. (Logophiles might appreciate the fact that these *free-form* requests are a result of the system being *form-free*.)

The easier it is to create forms, the more likely they will be created. Sadly, this feature is typically locked down so only high priests of the system may create forms. I think it's best to make it easy for anyone not only to suggest a form, but also to create one.

Github.com makes creating forms as easy as creating a text file. I know this because many open source projects have created forms to guide users through the bug-reporting process. It's amazing how many community-driven software projects have better practices than industry.

Rather than having many specialized forms, some helpdesk systems have one monster e-form that changes dynamically as the user is guided through a series of questions. These never work well for me. I find my request either does not fit into their categories, or the categories are proxies for an internal org chart that is unknown to the end user, or the system insists on requesting irrelevant information. Years ago, I found myself using such a system to request the facilities department replace a dead fluorescent light in my office. I had to list which operating system I was using and whether the problem could be reproduced on the previous version of "the software," whatever that could possibly mean.

You can tell if your helpdesk isn't providing the right forms when the typical first reply from any ticket is a list of information needed to proceed. "Thank you for submitting your request. To proceed, reply with the name of your computer." This adds an additional "round trip" before you can get the assistance you need. This delay could be prevented with a simple form.

Some helpdesk systems connect to inventory and personnel databases to reduce the amount of information humans must collect. For example, why ask a user for their phone number when it is in the corporate directory? Why ask for their computer's name when it could ask the user to choose from a list of host names associated with the user in the inventory database?

Not all IT professionals work at a helpdesk, but most work in a system where they receive transactional requests from others. Likewise, there are plenty of ways to convey to coworkers, partners, and customers what information is required to perform a particular request. Where I work, we have an internal Stack Overflow for Teams instance. I assure you that many procedural questions have answers that begin with a list of information to provide to assure smooth sailing.

Example 3: A Pre-engagement Questionnaire

While some software as a service (SaaS) products can be activated in a self-service manner, more complex ones require an onboarding process where a "customer success team" helps with initial setup. Typically, the customer is asked to answer several questions, usually in the form of a Microsoft Excel spreadsheet. The questions include the company's domain name, various security-related settings, and so on.

Process improvement is everyone's job, and when managers create roadblocks that prevent the people doing the work from making improvements, they do a disservice to the company.

Recently, I was involved in a project where the questionnaire had grown quite long. Originally, the questionnaire was a godsend—the chaos of emails flying around was replaced by a spreadsheet. Over time, however, the number of questions increased. New business requirements led to more questions. Each time a customer was permitted even the most obscure customization, that customization would soon be offered via the questionnaire. The logic was if one customer needed something, future customers might need it, too.

Thus, the questionnaire had become long, complex, and full of obscure technical terminology that not every customer could decipher.

Now this godsend was becoming a scourge.

Where it once took minutes to complete the questionnaire, it was now taking customers a week or more. The more questions, the more people involved in completing the form—and each new person added delays.

The other problem was an increase in requests for customizations that were difficult and laborious to implement. Yes, we can provide some obscure HTTP technobabble feature, but it requires a subject matter expert to step in and manually engineer it, delaying setup. What had been a special request from one or two customers with rare and unique requirements was now a "feature" visible to all customers.

Many customers would feel compelled to find technical subject matter experts who understood the question. Finding the right internal person could take days—so much fussing just to realize the feature was not needed. Salespeople would beg customers, "If you don't understand the question, go with the default," but that didn't help. In enterprise computing, no option is left unconsidered. Worst of all, customers who did not need such features would request them more out of curiosity than need.

Many items in the questionnaire had become obsolete and should have been removed or rewritten. Unfortunately, there was no process for removing questions.

Restructuring the questionnaire. Eventually, the team reworked the questionnaire. They divided it into three distinct sections: required answers, optional answers, and the secret menu.

The first section contained information that was always required. Lacking any of the required items was a blocker for the customer success team. When all was said and done, it was surprising how few questions remained in this section. Many questions could be moved out of this section once someone asked, "Are you sure it's really needed?"

For example, as is the case for many SaaS products, the customer's company logo is displayed in various UI elements such as the top navigation header. Conventional wisdom is this logo is required. How could we possibly launch a new customer's installation without it displaying their logo? It turns out, however, that is not a blocking requirement. In the absence of a customer-provided logo, a default generic icon is displayed. The customer could easily change the logo post-setup via the control panel on their own schedule. It was *not* a blocker.

So, the question was moved out of the "blocker list" and rephrased to say "the logo is to be installed by the customer post-setup, but if you happen to have it now, attach it here and we will take care of it for you as part of the initial setup." This sounds like a minor thing, but it was a big hurdle. At some companies, anything having to do with using the company logo requires weeks of approvals. You must meet with marketing, get approval from brand management, receive sign-off from legal, get a note from the CEO's mother, and more, depending on the company. These approvals were a common source of delay. This delay, however, was removed because someone pushed back and asked, "Are you really sure the logo is a blocker?"

The second section of the questionnaire was a list of "typical options." The team revised the questions to be clearer and more concise. All of them were constructed so the default was the preferred option. Someone could leave the section untouched and still be a satisfied customer.

A few of the questions originally targeted for this section were removed when we realized they reflected what was contractually obligated. We do not need to ask how frequently a report is generated if the contract requires us to generate it every 30 days. The customer success team had been prefilling these fields to reflect what was in the contract, but that did not stop customers from trying to make changes. The ensuing confusion wasted time for everyone.

The remaining questions were for extremely rare options. Most of them required manual engineering to implement. While we could fulfill these requests, we wanted to discourage them.

Originally, we thought we would list them in a section with a disclaimer that discouraged customers from selecting them: "Please use the default unless you have a special need," but that seemed rather unsociable.

Instead, we adopted a model used by some restaurants. Certain hamburger chains have a "secret menu." These are items that do not appear on the menu, but if you ask for one, your server will give you a knowing nod and deliver it to you. (A Web search for "restaurant secret menu" may surprise you with how common this is).

The remaining questions were removed from the questionnaire and placed on the "secret menu." Employees were trained not to offer these customizations unless the customer raised the issue. The secret menu listed alternatives that could solve the customer's problem in ways that the product could more easily support. "You need X? Would Y suffice because it is built into the product?" If the alternatives did not suffice, the request would be granted, although approval was required in some cases.

Restructuring the questionnaire into must-haves and options, plus having a secret menu, has greatly improved the team's ability to set up new customers in a predictable and trouble-free manner.

Summary and Conclusion

Knowing what information you need before beginning a task, and having that information, is key to productivity.

On an individual level, it is the difference between Bertie completing a task quickly and Andrew requiring multiple days.

On a team level, it allows a service desk to start on a request right away, and not be delayed by many round trips of clarifying questions.

On an organizational level, it streamlines and clarifies complicated processes, improves communication, reduces confusion, provides a focal point for optimizing processes, and looks much more professional.

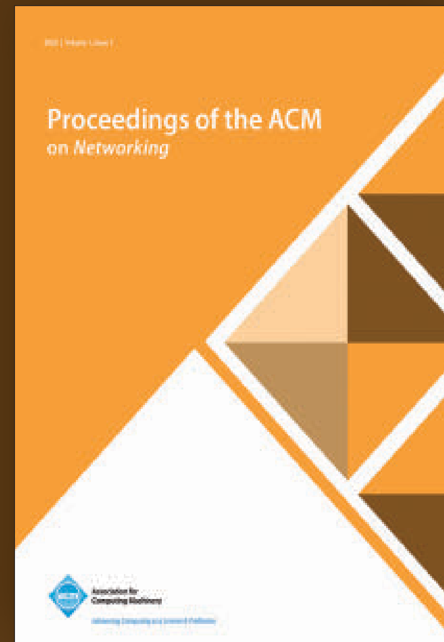
An analogy can be made to functional programming languages or anything that encourages code to be free of side effects. In simple terms, the easiest way to reduce side effects in a procedure is to assure it has all the data it needs passed to it from the start. This often requires a little foresight and taking the time to structure, and possibly refactor, code to make this easier to achieve. Similarly, whether you seek to improve on a personal, team, or organizational level, it can be valuable to pause, reflect, and refactor, so you have all the information needed before you begin.

Gathering all required information ahead of time is not always possible. A task may be ill-defined or require ad hoc solutions. Even then, you can ensure the knowable elements are known before you begin.

The better you get at this, the more you can be like Bertie. ■

Thomas A. Limoncelli is a technical product manager at Stack Overflow Inc. He previously worked at small and large companies, including Google, Bell Labs/Lucent, and AT&T. He blogs at EverythingSysadmin.com and posts at @YesThatTom.

OPEN FOR SUBMISSIONS



Proceedings of the ACM on Networking (PACMNET)

Editors-in-Chief

Marco Mellia
Politecnico di Torino, Italy

Peter Steenkiste
Carnegie Mellon University, USA

Publishing significant and novel research results on emerging computer networks and its applications

Proceedings of the ACM on Networking (PACMNET) is a journal for research relevant to multiple aspects of the area of computer networking. Submissions that present new technologies, novel experimentation, creative use of networking technologies, and new insights made possible using analysis are especially encouraged.

PACMNET invites submissions on a wide range of networking topics, including: protocols in different types of networks (public Internet, data center networks, home and enterprise networks, sensor networks, wireless networks), network measurements and modeling, evaluation of networks and networked systems using diverse techniques (verification and modeling, trace driven simulation, testbed, in-the-wild experiments), experience and lessons learned from deployments of networks and their applications, network management and control, including routing, traffic engineering, SDN, NFV, network programmability, and applications of machine learning in computer networking. We particularly welcome experimental results, and papers offering additional artifacts such as code, datasets, etc., in the light of reproducibility.

For more information and to submit your manuscript, please visit pacmnet.acm.org.

Association for Computing Machinery

DOI:10.1145/3624700

AI fairness should not be considered a panacea: It may have the potential to make society more fair than ever, but it needs critical thought and outside help to make it happen.

BY MAARTEN BUYL AND TIJL DE BIE

Inherent Limitations of AI Fairness

IN THE PAST, the range of tasks that a computer could carry out was limited by what could be hard-coded by a programmer. Now, recent advances in machine learning (ML) make it possible to learn patterns from data, such that we can efficiently automate tasks where the decision process is too complex to manually specify. After sensational successes in computer vision and natural language processing (NLP), the impact of artificial intelligence (AI) systems powered by ML is rapidly widening toward other domains.

AI is already being used to make high-stakes decisions in areas such as credit scoring,[40] predictive policing,[31] and employment.[20] These are highly sensitive applications, in which decisions must be *fair*; that is, non-discriminatory with respect to an individual's *protected traits* such as gender, ethnicity, or religion. Indeed, the principle of non-discrimination has a long history and intersects with many viewpoints: as a value in moral philosophy,[5] as a human right,[13] and as a legal protection.[44] Evidently, algorithms that play a role in such decision processes should meet similar expectations, which is why *AI fairness* is prominently included as a value in the AI ethics guidelines of many public- and private-sector organizations.[27] Yet, AI systems that blindly apply ML are rarely fair in practice, to begin with because training data devoid of undesirable biases is hard to come by.[26] At the same time, fairness is difficult to hard-code, because it demands nuance and complexity.

Hence, fairness is both important and difficult to achieve; as such it is high on the AI research agenda.[36] The dominant approach to AI fairness in computer science is to formalize it as a mathematical constraint, before imposing it upon the decisions of an AI system while losing as little predictive accuracy as possible. For example, assume that we want to design an AI system to score people who apply to enroll at a university. To train this AI system, we may want to learn from data of past decisions made by human adminis-

» key insights

- **The field of AI fairness aims to measure and mitigate algorithmic discrimination, but the technical formalism this requires has come under increasing criticism in recent years.**
- **In the prototypical, technical approach to AI fairness, eight inherent limitations can be identified that inhibit its potential to truly address discrimination in practice. These include concerns about the assessment of performance, the need for sensitive data, and the limited power of technical formalization in high-impact decision processes.**
- **The limitations delineate the role technical tools for fairness should play and serve as a disclaimer for their current paradigm.**

ILLUSTRATION BY PETER CROWTHER ASSOCIATES

trators in the admission process. Yet, if those decisions consistently undervalued women in the past, then any system trained on such data is likely to inherit this bias. In the prototypical fair classification setting, we could formalize fairness by requiring that men and women are, on average, admitted at equal proportions. A simple technical AI fairness approach to meet this constraint is to uniformly increase the scores given to women. The resulting decisions may then be considered fair, and little accuracy is likely to be lost.

Yet this approach rests on simplistic assumptions. For instance, it assumes that fairness can be mathematically formalized, that we collect gender labels on every applicant, that everyone fits in a binary gender group, and that we only care about a single axis of discrimination. The validity of such assumptions has recently been challenged, raising concerns over technical interventions in AI systems as a panacea for ensuring fairness in their application.[10,34,38]

Despite this criticism, the idea of tackling fairness through mathematical formalism remains popular. To an extent, this is understandable. Large-scale data collection on humans making high-impact decisions could enable us to study biases in the allocation of resources with a level of statistical power that was previously infeasible. If we could properly mitigate biases in AI systems, then we may even want fair AI to replace these human decision makers, such that the overall fairness of the decision process is improved. At the very least, technical solutions in fair AI seem to promise pragmatic benefits that are worth considering.

Therefore, the aim of this article

is to estimate how far (technical) AI fairness approaches can go in truly measuring and achieving fairness by outlining what *inherently* limits it from doing so in realistic applications. With this lens, we survey criticisms of AI fairness and distill eight such inherent limitations. These limitations result from shortcomings in the assumptions on which AI fairness approaches are built. Hence, they are considered fundamental, practical obstacles, and we will not frame them as research challenges that can be solved within the strict scope of AI research. Rather, our aim is to provide the reader with a disclaimer for the ability of fair AI approaches to address fairness concerns. By carefully delineating the role that it can play, technical solutions for AI fairness can continue to bring value, though in a nuanced context. At the same time, this delineation provides research opportunities for non-AI solutions peripheral to AI systems, and mechanisms to help fair AI fulfill its promises in realistic settings.

Remark: *In what follows, we start from the technical perspective on AI fairness, such that we can test its inherent limits and their relation with socio-technical, legal, and industry challenges. Related work takes a different approach by arguing that, since fairness is fundamentally a socio-technical concept, we should anyhow not expect technical tools to be sufficient in pursuing fairness.[38] The significance of technical tools should instead be found in the informational role they play in service of a wider socio-technical discussion.[1] The present article is written from a different perspective, thus corroborating this conclusion.*

Limitations of AI Fairness

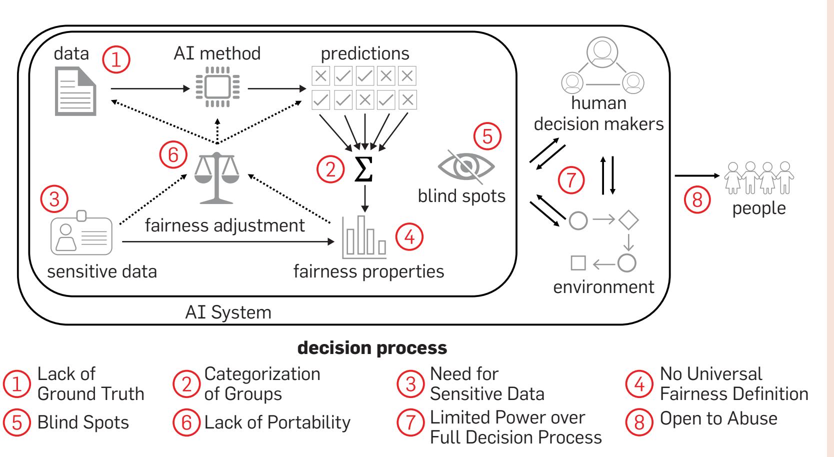
Figure 1 provides an overview of the prototypical AI fairness solution.[36] In this setting, an AI method learns from data, which may be biased, to make predictions about individuals. Task-specific fairness properties are computed by categorizing individuals into *protected groups*, such as *men* and *women*, and then comparing aggregated statistics over the predictions for each group. Without adjustment, these predictions are assumed to be biased, because the algorithm may inherit bias from the data and because the algorithm is probably imperfect and may then make worse mistakes for some of the groups. Fairness adjustment is done using *preprocessing* methods that attempt to remove bias from the data, *inprocessing* methods where modifications are made to the learning algorithm such that discriminatory patterns are avoided, and *postprocessing* methods that tweak the predictions of a potentially biased learning algorithm.

We consider eight inherent limitations of this prototypical fair AI system, which each affect its various components and levels of abstraction as illustrated in Figure 1. To start, we observe that bias in the data results in biased approximations of the *ground truth*, leading to unfair conclusions about the performance and fairness properties of the AI system. Fairness measurements are also problematic because they involve distinguishing people into groups and require *sensitive data* of individuals to do so. In fact, there is generally no universal definition of fairness in exact terms. Avoiding *any* discrimination is anyhow unrealistic, as it demands that the possible sources of bias are well-understood without any blind spots. Even if fairness can be adequately elaborated for one setting, the same solutions will often not be portable to another. More generally, AI systems are only a component of a larger decision process, for example, where biases can arise in the interaction with human decision makers and the environment, that is no longer within the scope of the AI. In this larger scope, we conclude that AI fairness can be abused, through negligence or malice, to worsen injustice.

Lack of ground truth. As is now widely understood by the AI fairness community, datasets that involve humans are often plagued by a range of biases that may lead to discrimination in algorithmic decisions.[36] Importantly, this bias may prevent an unbiased estimation of the ground truth—that is, the prediction that should have been made for an individual. It is then difficult to make unbiased statements about the performance of an algorithm.[10] For example, in predictive policing, when trying to predict whether someone will commit a crime, the dataset may use 'arrested' as the output label that must be predicted. Yet, when one group is policed more frequently, their arrest rates may not be comparable to an advantaged group, meaning it cannot be assumed to be an unbiased estimate of the ground truth 'committed a crime'.[14] This raises concerns whether predictive policing can ever be ethical after long histories of biased practices.

In general, the lack of a ground truth is a significant problem for fair AI, as it often depends on the availability of the ground truth to make statements about fairness. For instance, a large disparity between false positive rates formed the basis of criticism against the COMPAS algorithm, which was found to incorrectly predict high recidivism risk for black defen-

Figure 1. A prototypical fair AI system. Each limitation affects a different component of the full decision process.



dants more often than for white defendants.[31] However, lacking a ground truth to compute those rates in an unbiased way, one cannot even measure algorithmic fairness in this setting.[21] Caution is thus warranted in the interpretation of any metrics where ground truth should be used to compute them. This holds for overall performance metrics such as accuracy, but also for many fairness statistics.

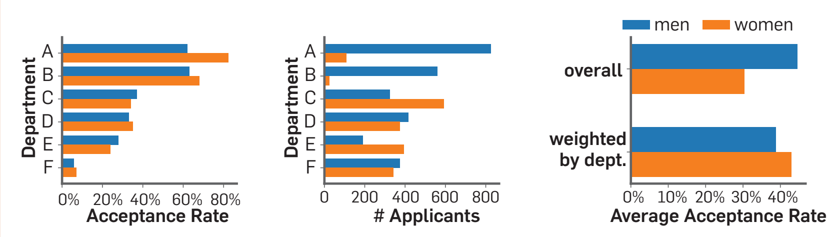
Categorization of groups. Bias has been considered in algorithms since at least 1996. Yet, it was the observations that biases are found throughout real datasets and then reproduced by data-driven AI systems that led to the rapid development of AI fairness research.[4] AI systems are typically evaluated and optimized with a formal, mathematical objective in mind, and so this field demands formalizations of fairness that somehow quantify discriminatory biases in exact terms.

Assuming the AI system is meant to classify or score individuals, a strong formalization of fairness is the notion of *individual* fairness, which asserts that "any two individuals who are similar with respect to a particular task should be classified similarly."[15] Though principled, such definitions require an unbiased test to assess whether two individuals are indeed similar. However, if such a test were readily available, then we could directly use that test to construct an unbiased classifier. Developing a test for strong definitions of individual fairness is thus equally as hard as solving the initial problem of learning a fair classifier.

The vast majority of AI literature is instead concerned with the more easily attainable notion of *group fairness*, which requires that any two protected groups should, on average, receive similar labels.[36] Group fairness expresses the principles of individual fairness[6] by looking at the sum of discrimination toward an entire group rather than individual contributions. Though this increased statistical power makes group fairness much more practical to measure and satisfy, it comes with its own problems, which we discuss next.

Simpson's paradox. An obvious problem with group fairness is that, through aggregation, some information will be lost. To illustrate, we discuss the example of the University of California, Berkeley admission statistics from the fall of 1973.[19] These statistics showed that the overall acceptance rate for men was higher (44%) than for women (35%), suggesting there was a pattern of discrimination against the latter. However, when we consider the acceptance rates separately per department[a] in Figure 2, there does not seem to be such a pattern. In fact, except for departments C and D, the acceptance rate is reportedly higher for women. From this perspective, it could be argued that women are favored. The fact that conclusions may vary depending on the granularity of the groups is referred to as *Simpson's Paradox*.

Figure 2. UC Berkeley admissions statistics for men and women. Left: Acceptance rates. Middle: Number of applicants. Right: Average acceptance rate, either overall or weighted by the total number of applicants (of both groups) for each department.



The paradox arises from the difference in popularity between the departments. Women more commonly applied to departments with low acceptance rates (for example, departments C through F), which hurt their *overall* average acceptance rate. Yet, when taking the *weighted* average by accounting for the total number of applicants of both men and women, we see that women score higher. As noted by Wachter et al.,[44] the point of this example is not that some forms of aggregation are better than others. Rather, we should be wary that aggregation, such as the kind that is found in group fairness, may influence the conclusions we draw.

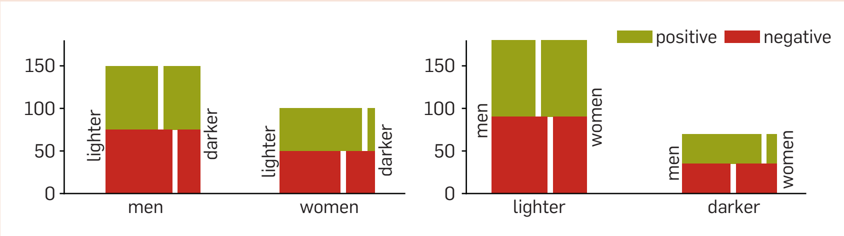
Fairness gerrymandering. An especially worrisome property of group fairness is that, by looking at averages over groups, it allows for some members of a disadvantaged group to receive poor scores (for example, due to algorithmic discrimination), so long as other members of the same group receive high-enough scores to compensate. Such situations are considered *fairness gerrymandering*[28] if specific subgroups can be distinguished as systematically disadvantaged within their group.

Group-fairness measures may thus be hiding some forms of algorithmic discrimination, illustrated by the toy example in Figure 3. It is constructed such that *men* and *women* receive positive decisions at equal average rates (here 50%), just like when we view the groups of *lighter-skinned* and *darker-skinned* people separately. A system giving such scores might therefore be seen as fair, because group fairness measures no discrepancy based on gender or skin tone. However, this per-axis measurement hides the discrimination that may be experienced at the intersection of both axes. In the case of Figure 3, lighter-skinned men and darker-skinned women proportionally receive fewer positive predictions than the other intersectional groups. Because darker-skinned women are also in the minority, they receive even fewer positive predictions proportionally. Consider, then, that in the real world, we often do see significant discrimination along separate axes. The discrimination experienced by those at the intersection may then be far worse. Indeed, empirical evidence shows, for example, that darker-skinned women often face the worst error rates in classification tasks.[8]

The field of *intersectional fairness* addresses the (magnified) discrimination experienced by those at the intersection of protected groups.[8] A straightforward approach is to directly require fairness for all combinations of subgroups. While this addresses some of the concerns for the example in Figure 3, it is not a realistic solution. Such definitions of intersec-

a Note that most departments were omitted in the example.

Figure 3. Toy example where fairness along two axes separately does not imply fairness along both axes simultaneously. Each vertical bar shows the number of positives and negatives for a group seen along the axis of either gender (left) or skin tone (right) separately. Each bar is also split in two such that the proportions for each intersectional subgroup (for example, darker-skinned women) is visible.



tional fairness give rise to a number of subgroups that grows exponentially with the number of axes of discrimination,[28] thereby losing the statistical power we were hoping to gain with group fairness.

Discretization concerns. Sociotechnical views of intersectional fairness also criticize the assumption that people can neatly be assigned to groups at all. This assumption helps in mapping legal definitions of discrimination to a technical context, yet it thereby also inherits the issues that result from rigid identity categories in law.[25] In fact, most protected characteristics resist discretization. Gender and sexual orientation are unmeasurable and private,[37] disability is highly heterogeneous,[9] and ethnicity labels are subjective and may be reported differently depending on the time and place.[34]

While the fear of gerrymandering in group fairness encourages highly granular (intersections of) identity groups, an 'overfitting' of identity categorization should also be avoided. It must be better understood how machine learning, which aims at generalization, can model such individual nuances.

Need for sensitive data. Fair AI research is fueled by the use of sensitive data—that is, data on people's traits that are protected from discrimination. Almost all methods require such data to measure and mitigate bias.[36] Indeed, exact fairness definitions involve categorizing people into groups, which requires us to know, for every person, which group they actually identify with (if any).

Yet it is hardly a viable assumption that sensitive data is generally available for bias mitigation purposes in the real world. Clearly, the collection of such highly personal data conflicts with the ethical principle of privacy,[42] despite the fact that privacy and fairness are often considered joint objectives in ethical AI guidelines.[27] Fairness should not blindly be given priority here, since disadvantaged groups may prefer to avoid disclosing their sensitive data due to distrust caused by structural discrimination in the past.[2] For example, outing the sexual orientation or gender identity of queer individuals can lead to emotional distress and serious physical or social harms.[41]

Indeed, practitioners report that individuals are often hesitant to share sensitive data, most frequently because they do not trust that it will be in their benefit.[3] Privacy and availability concerns are thus consistently cited as significant obstacles to implementing fair AI in practice.[26,32,37] Sensitive data has received special protection in the past, such as in European data protection law.[b] In stark contrast, the European Commission's recently proposed AI Act[c] now includes a provision that specifically allows for the processing of sensitive data for the purpose of ensuring "the bias monitoring, detection and correction in relation to high-risk AI systems."

No universal fairness definition. So far, we have discussed mathematical definitions of fairness in simple terms—that is, as constraints that compare simple statistics of different demographic groups. These constraints are motivated through the idea that the AI system is distributing resources, which ought to be done in a *fair* way. However, does this mean that each group should receive an *equal* amount of resources, or should we try to be *equitable* by taking the particular need, status, and contribution of these groups into account?[45] This irresolution has led to a notoriously large variety of fairness notions. For just the binary classification setting, one survey from 2018[43] lists 20 different definitions, each with its own nuance and motivations. For example, *statistical parity*[15] requires equality in the rate at which positive predictions are made. On the other hand, *equalized odds* requires *false positive* and *false negative rates* to be equal. It therefore allows for different rates at which groups receive positive decisions, so long as the amount of mistakes is proportionally the same. Both notions are defensible depending on the context, but they are also generally incompatible: If the base rates differ across groups (that is, one group has a positive label in the data more often than another), then statistical parity and equalized odds can only be met simultaneously in trivial cases.[29]

In fact, many notions have shown to be mathematically incompatible in realistic scenarios.[21] This has led to controversy, for example, in the case of the COMPAS algorithm. As mentioned previously, it displayed a far higher false-positive rate in predicting recidivism risk for black defendants than white defendants.[31] Yet the COMPAS algorithm was later defended by noting that it was equally calibrated for both groups; that is, black and white defendants that were given the same recidivism risk estimate indeed turned out to have similar recidivism rates.[17] Later studies proved that such a balance in calibration is generally incompatible with a balance in false-positive and negative rates, except in trivial cases.[29] From a technical point of view, these results imply that there cannot be a single definition of fairness that will work in all cases. Consequently, technical tools for AI fairness should be flexible in how fairness is formalized, which greatly adds to their required complexity.

Moreover, it may not even be possible to find a 'most relevant' fairness

b Most notably, see the General Data Protection Regulation: https://bit.ly/3RGnqKR.

c At press time, the AI Act is still being debated. The most recent milestone is the set of amendments adopted by the European Parliament: https://bit.ly/47eS7wb definition for a specific task. Indeed, human-subject studies show that humans generally do not reach a consensus on their views of algorithmic fairness. From studies collected in a surveyed by Starke et al.,[39] it appears that people's perceptions of fairness are heavily influenced by whether they personally receive a favorable outcome. The 'most relevant' fairness definition may therefore be the product of a constantly shifting compromise[7] resulting from a discussion with stakeholders. It is also noted that these discussions may be hampered by a lack of understanding of technical definitions of fairness.

Blind spots. It will now be clear that fairness is inherently difficult to formalize and implement in AI methods. We thus stress the need for nuance and iteration in addressing biases. However, such a process assumes that possible biases are already anticipated or discovered. Businesses have raised concerns that, though they have processes and technical solutions to tackle bias, they can only apply them to biases that have already been foreseen or anticipated.[32] Without an automated process to find them, they are limited to following hunches of where bias might pop up. Holstein et al.[26] cite one tech worker, saying, "You'll know if there's fairness issues if someone raises hell online." Though well-designed checklists may help to improve the anticipation of biases,[35] it is safe to assume that some blind spots will always remain. Since unknown biases are inherently impossible to measure, we cannot always make definitive guarantees about fairness in their presence. The fairness of AI systems should thus constantly be open to analysis and criticism, such that new biases can quickly be discovered and addressed.

In fact, some biases may only arise *after* deployment. For example, the data on which an algorithm is trained may have different properties than the data on which it is evaluated, because the distribution of the latter may be continuously changing over time.[30] Similarly, any fairness measurements taken during training may not be valid after deployment.[12] The AI Act proposed by the European Commission would therefore rightly require that high-risk AI systems are subjected to post-market monitoring, in part to observe newly arising biases. The fairness properties of an AI system should thus continuously be kept up to date.

In an unjust world, it is meaningless to talk about the fairness of an algorithm's decisions without considering the wider socio-technological context in which an algorithm is applied.

Lack of portability. As we have argued, fairness should be pursued for a particular AI system after carefully elaborating the assumptions made and after finding compromise between different stakeholders and viewpoints—for example, because fairness is difficult to formalize. However, this means that our desiderata for a fair AI system become highly situation-specific. As pointed out by Selbst et al.,[38] this limits the portability of fairness solutions across settings, even though portability is a property that is usually prized in computer science. Consequently, the flexibility of fair AI methods that we called for earlier in this article may not be achievable by simply choosing from a zoo of portable fairness solutions.

In industry, there is already empirical evidence that off-the-shelf fairness methods have serious limits. For example, fairness toolkits such as *AIF360*,[d] *Aequitas*,[e] and *Fairlearn*[f] offer a suite of bias measurement, visualization, and mitigation algorithms. However, though such toolkits can be a great resource to learn about AI fairness, practitioners found it hard to actually adapt them to their own model pipeline or use case.[33]

Limited power over the full decision process. Fair AI papers often start by pointing out that algorithms are increasingly replacing human decision makers. Yet, decisions in high-risk settings, such as credit scoring, predictive policing, or recruitment, are expected to meet ethical requirements. Such decision processes should thus only be replaced by algorithms that meet or exceed similar ethical requirements, such as fairness. However, it is unrealistic to assume that decision processes will be fully automated in precisely those high-risk settings that motivate fair AI in the first place. This is because fully automated AI systems are not trusted to be sufficiently accurate and

d See https://github.com/Trusted-AI/AIF360
e See https://github.com/dssg/aequitas
f See https://github.com/fairlearn/fairlearn

fair.[7] The EU's General Data Protection Regulation[g] even specifically grants the right to, in certain circumstances, *not* be subject to fully automated decision making.

In most high-risk settings, algorithms only play a supportive role and the final decision is subject to human oversight.[22] However, unfairness may still arise from the interaction between the algorithm and the larger decision process (for example, including human decision makers) that is outside its scope.[38] For example, a study conducted by Green and Chen[23] asked participants to give a final ruling on defendants in a pre-trial setting. After being presented with an algorithmic risk assessment, participants tended to (on average) assign a higher risk to black defendants than the algorithm. The reverse was true for white defendants.

Hence, in an unjust world, it is meaningless to talk about the fairness of an algorithm's decisions without considering the wider socio-technological context in which an algorithm is applied.[16] Instead, it is more informative to measure the overall fairness properties of a decision process, of which an algorithm may only be a small component among human decision makers and other environmental factors.[11]

Open to abuse. Like most technology, solutions for algorithmic fairness are usually evaluated with the understanding that they will be used in good faith. Yet the sheer complexity of fairness may be used as a cover to avoid fully committing to it in practice. Indeed, opportunistic companies implementing ethics into their AI systems may resort to *ethics-shopping*—that is, they may prefer and even promote interpretations of fairness that align well with existing (business) goals.[18] Though they may follow organizational 'best practices' by establishing ethics boards and collecting feedback from a wide range of stakeholders, an actual commitment to moral principles may mean doing more than what philosophers are allowed to do within corporate settings.[5] Fundamentally, solutions toward ethical AI have a limited effect if a deviation from ethics has no consequences.[24]

The sheer complexity of fairness may be used as a cover to avoid fully committing to it in practice.

The complexity of fairness may not only lead to obscure commitments toward it; its many degrees of freedom may also be abused to create or intensify injustice. In this article, we argued that ground truth labels are often unavailable, lending power to whomever chooses the proxy that is used instead. Moreover, as we also discussed, the groups in which people are categorized grant the power to conceal discrimination against marginalized subgroups. Following another thread in this article on the need for sensitive data to measure fairness, we also warn that this necessity could motivate further surveillance, in particular on those already disadvantaged.[10] Overall, significant influence is also afforded in deciding how fairness is actually measured, as a universal definition may not exist.

Conclusion

It is evident from our survey of criticism toward fair AI systems that such methods can only make guarantees about fairness based on strong assumptions that are unrealistic in practice. Hence, AI fairness suffers from inherent limitations that prevent the field from accomplishing its goal on its own. Some technical limitations are inherited from modern ML paradigms that expect reliable estimates of the ground truth and clearly defined objectives to optimize. Other limitations result from the necessity to measure fairness in exact quantities, which requires access to sensitive data and a lossy aggregation of discrimination effects. The complexity of fairness means that some forms of bias will always be missed and that every elaboration of fairness is highly task-specific. Moreover, even a perfectly designed AI system often has limited power to provide fairness guarantees for the full decision process, as some forms of bias will remain outside its scope. Finally, the extensive automation of high-stakes decision processes with allegedly fair AI systems entails important risks, as the complexity of fairness opens the door to abuse by whomever designs them.

These inherent limitations motivate why AI fairness should not be

g See https://bit.ly/3RGnqKR

more online

A list of full references and supplementary information is available at https://dl.acm.org/doi/10.1145/3624700.

considered a panacea. Yet, we also stress that the many benefits of AI fairness must not be overlooked, since it can remain a valuable tool as part of broader solutions. In fact, many of the limitations we identify and assumptions we question are not only inherent to fair AI, but to the ethical value of fairness in general. The study of AI fairness thus forces and enables us to think more rigorously about what fairness really means to us, lending us a better grip on this elusive concept. In short, fair AI may have the potential to make society more fair than ever, but it needs critical thought and outside help to make it happen.

Acknowledgments

This research was funded by the ERC under the E.U.'s 7th Framework and H2020 Programmes (ERC Grant Agreement no. 615517 and 963924), the Flemish Government (AI Research Program), the BOF of Ghent University (BOF20/DOC/144 and BOF20/IBF/117), and the FWO (G0F9816N, 3G042220).

References

1. Abebe, R. et al. Roles for computing in social change. In *Proceedings of the 2020 Conf. on Fairness, Accountability, and Transparency,* Association for Computing Machinery, 252–260; 10.1145/3351095.3372871
2. Andrus, M., Spitzer, E., Brown, J., and Xiang, A. What we can't measure, we can't understand: Challenges to demographic data procurement in the pursuit of fairness. In *Proceedings of the 2021 ACM Conf. on Fairness, Accountability, and Transparency*. ACM, 249–260; 10.1145/3442188.3445888
3. Andrus, M. and Villeneuve, S. Demographic-reliant algorithmic fairness: Characterizing the risks of demographic data collection in the pursuit of fairness. In *Proceedings of the 2022 ACM Conf. on Fairness, Accountability, and Transparency*, 1709–1721; 10.1145/3531146.3533226
4. Barocas, S., Hardt, M., and Narayanan, A. Fairness and machine learning: Limitations and opportunities. *fairmlbook.org* (2019).
5. Bietti, E. From ethics washing to ethics bashing: A view on tech ethics from within moral philosophy. In *Proceedings of the 2020 Conf. on Fairness, Accountability, and Transparency*. Association for Computing Machinery, 210–219; 10.1145/3351095.3372860
6. Binns, R. On the apparent conflict between individual and group fairness. In *Proceedings of the 2020 Conf. on Fairness, Accountability, and Transparency*. ACM, 514–524; 10.1145/3351095.3372864
7. Binns, R. et al. 'It's reducing a human being to a percentage': Perceptions of justice in algorithmic decisions. In *Proceedings of the 2018 CHI Conf. on Human Factors in Computing Systems*. Association for Computing Machinery, 1–14; 10.1145/3173574.3173951
8. Buolamwini, J. and Gebru, T. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Proceedings of the 1st Conf. on Fairness, Accountability and Transparency*. PMLR (2018), 77–91.
9. Buyl, M., Cociancig, C., Frattone, C., and Roekens, N. Tackling algorithmic disability discrimination in the hiring process: An ethical, legal and technical analysis. In *2022 ACM Conf. on Fairness, Accountability, and Transparency*, Association for Computing Machinery, 1071–1082; 10.1145/3531146.3533169
10. Feder Cooper, A. and Abrams, E. Emergent unfairness in algorithmic fairness-accuracy trade-off research. In *Proceedings of the 2021 AAAI/ACM Conf. on AI, Ethics, and Society*. Association for Computing Machinery, 46–54.
11. Cortés, E.C., Rajtmajer, S., and Ghosh, D. Locality of technical objects and the role of structural interventions for systemic change. In *2022 ACM Conf. on Fairness, Accountability, and Transparency*. Association for Computing Machinery, 2327–2341; 10.1145/3531146.3534646
12. D'Amour, A. et al. Fairness is not static: Deeper understanding of long term fairness via simulation studies. In *Proceedings of the 2020 Conf. on Fairness, Accountability, and Transparency*. Association for Computing Machinery, 525; 10.1145/3351095.3372878
13. Donahoe, E. and MacDuffee Metzger, M. Artificial intelligence and human rights. *J. of Democracy 30*, 2 (2019), 115–126; 10.1353/jod.2019.0029
14. Dressel, J. and Farid, H. The accuracy, fairness, and limits of predicting recidivism. *Science Advances 4*, 1 (Jan. 2018); 10.1126/sciadv.aao5580
15. Dwork, C. et al. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conf.*, Association for Computing Machinery (2012), 214; 10.1145/2090236.2090255
16. Fazelpour, S. and Lipton, Z.C. Algorithmic fairness from a non-ideal perspective. In *Proceedings of the AAAI/ACM Conf. on AI, Ethics, and Society*. Association for Computing Machinery (2020), 57; 10.1145/3375627.3375828
17. Flores, A.W., Lowenkamp, C.T., and Bechtel, K. False positives, false negatives, and false analyses: A rejoinder to "machine bias: There's software used across the country to predict future criminals. And it's biased against blacks." *Federal Probation J. 80*, 2 (Sept. 2016).
18. Floridi, L. Translating principles into practices of digital ethics: Five risks of being unethical. *Philosophy & Technology 32*, 2 (June 2019), 185–193; 10.1007/s13347-019-00354-x
19. Freedman, D., Pisani, R., and Purves, R. *Statistics: Fourth Intern. Student Edition*. W. W. Norton & Company (2007).
20. Noris Freire, M. and de Castro, L.N. E-recruitment recommender systems: A systematic review. *Knowledge and Information Systems 63*, 1 (Jan. 2021), 1–20; 10.1007/s10115-020-01522-8
21. Friedler, S.A., Scheidegger, C., and Venkatasubramanian, S. The (im)possibility of fairness: Different value systems require different mechanisms for fair decision making. *Communications of the ACM 64*, 4 (Mar. 2021), 136–143; 10.1145/3433949
22. Green, B. *The Flaws of Policies Requiring Human Oversight of Government Algorithms* (2022); 10.2139/ssrn.3921216
23. Green, B. and Chen, Y. The principles and limits of algorithm-in-the-loop decision making. In *Proceedings of the ACM on Human-Computer Interaction 3*, CSCW (Nov. 2019), 1–24; 10.1145/3359152
24. Hagendorff, T. The ethics of AI ethics: An evaluation of guidelines. *Minds and Machines 30*, 1 (Mar. 2020), 99–120; 10.1007/s11023-020-09517-8
25. Hoffmann, A.L. Where fairness fails: Data, algorithms, and the limits of antidiscrimination discourse. *Information, Communication & Society 22*, 7 (June 2019), 900–915; 10.1080/1369118X.2019.1573912
26. Holstein, K. et al. Improving fairness in machine learning systems: What do industry practitioners need? In *Proceedings of the 2019 CHI Conf. on Human Factors in Computing Systems,* Association for Computing Machinery, 1; 10.1145/3290605.3300830
27. Jobin, A., Ienca, M., and Vayena, E. The global landscape of AI ethics guidelines. *Nature Machine Intelligence 1*, 9 (Sept. 2019), 389–399; 10.1038/s42256-019-0088-2
28. Kearns, M., Neel, S., Roth, A., and Wu, Z.S. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. In *Proceedings of the 35th Intern. Conf. on Machine Learning,* PMLR (2018), 2564–2572.
29. Kleinberg, J., Mullainathan, S., and Raghavan, M. Inherent trade-offs in the fair determination of risk scores. In *Proceedings of th 8th Innovations in Theoretical Computer Science Conf., Leibniz Intern. Proceedings in Informatics 67* (2017), 1–23.
30. Wei Koh, P. et al. WILDS: A benchmark of in-the-wild distribution shifts. In *Proceedings of the 38th Intern. Conf. on Machine Learning*. PMLR (2021), 5637–5664.
31. Larson, J., Mattu, S., Kirchner, L., and Angwin, J. How we analyzed the COMPAS recidivism algorithm. *Propublica* (2016); https://bit.ly/3vbmMx3.
32. Law, P., Malik, S., Du, F., and Sinha, M. Designing tools for semi-automated detection of machine learning biases: An interview study. In *Proceedings of the CHI 2020 Workshop on Detection and Design for Cognitive Biases in People and Computing Systems*.
33. Lee, M.S.A. and Singh, J. The landscape and gaps in open source fairness toolkits. In *Proceedings of the 2021 CHI Conf. on Human Factors in Computing Systems*. Association for Computing Machinery, 1; 10.1145/3411764.3445261
34. Lu, C., Kay, J., and McKee, K. Subverting machines, fluctuating identities: Re-learning human categorization. In *Proceedings of the 2022 ACM Conf. on Fairness, Accountability, and Transparency*. ACM, 1005–1015; 10.1145/3531146.3533161
35. Madaio, M.A., Stark, L., Wortman Vaughan, J., and Wallach, H. Co-designing checklists to understand organizational challenges and opportunities around fairness in AI. In *Proceedings of the 2020 CHI Conf. on Human Factors in Computing Systems*. ACM, 1–14; 10.1145/3313831.3376445
36. Mehrabi, N. et al. A survey on bias and fairness in machine learning. *Comput. Surveys 54*, 6 (July 2021), 115:1–115:35; 10.1145/3457607
37. Rakova, B., Yang, J., Cramer, H., and Chowdhury, R. Where responsible AI meets reality: Practitioner perspectives on enablers for shifting organizational practices. In *Proceedings of the ACM on Human-Computer Interaction 5* (April 2021), 7:1–7:23; 10.1145/3449081
38. Selbst, A.D. et al. Fairness and abstraction in sociotechnical systems. In *Proceedings of the Conf. on Fairness, Accountability, and Transparency*. ACM (2019), 59–68; 10.1145/3287560.3287598
39. Starke, C., Baleis, J., Keller, B., and Marcinkowski, F. Fairness perceptions of algorithmic decision-making: A systematic review of the empirical literature. In *Proceedings of Big Data & Society 9*, 2 (July 2022); 10.1177/20539517221115189
40. Thomas, L., Crook, J., and Edelman, D. *Credit Scoring and Its Applications, Second Edition*. SIAM, (2017).
41. Tomasev, N., McKee, K.R., Kay, J., and Mohamed, S. Fairness for unobserved characteristics: Insights from technological impacts on queer communities. In *Proceedings of the 2021 AAAI/ACM Conf. on AI, Ethics, and Society*, 254–265; 10.1145/3461702.3462540
42. Veale, M. and Binns, R. Fairer machine learning in the real world: Mitigating discrimination without collecting sensitive data. *Big Data & Society 4*, 2 (Dec. 2017), 2053951717743530; 10.1177/2053951717743530
43. Verma, S. and Rubin, J. Fairness definitions explained. In *Proceedings of the Intern. Workshop on Software Fairness*. ACM (2018), 1–7; 10.1145/3194770.3194776
44. Wachter, S., Mittelstadt, B., and Russell, C. Why fairness cannot be automated: Bridging the gap between EU non-discrimination law and AI. *Computer Law & Security Rev. 41* (July 2021), 105567; 10.1016/j.clsr.2021.105567
45. Young, H.P. *Equity: In Theory and Practice*. Princeton University Press, (1995).

Maarten Buyl (maarten.buyl@ugent.be) is a postdoctoral researcher at Ghent University, Belgium.

Tijl de Bie is a professor at Ghent University, Belgium.

Watch the authors discuss this work in the exclusive *Communications* video. https://cacm.acm.org/videos/inherent-limitations

DOI:10.1145/3624720

Challenges and opportunities faced by computing educators and students adapting to LLMs capable of generating accurate source code from natural-language problem descriptions.

BY PAUL DENNY, JAMES PRATHER, BRETT A. BECKER, JAMES FINNIE-ANSLEY, ARTO HELLAS, JUHO LEINONEN, ANDREW LUXTON-REILLY, BRENT N. REEVES, EDDIE ANTONIO SANTOS, AND SAMI SARSA

Computing Education in the Era of Generative AI

A NEW ERA is emerging in which artificial intelligence (AI) will play an ever-increasing role in many facets of daily life. One defining characteristic of this new era is the ease with which novel content can be generated. Large language models (LLMs)—neural network-based models trained on vast quantities of text data[4]—are capable of creating a variety of convincing human-like outputs, including prose, poetry, and source code. It is largely accepted that synthesizing source code automatically from natural-language prompts is likely to improve the productivity of professional developers[26] and is being actively explored by well-funded entities such as OpenAI (ChatGPT, GPT-4[a]), Amazon (CodeWhisperer[b]), and Google (Alpha-Code,[21] Bard[c]). In the same way that high-level programming languages offered large productivity advantages over assembly-language programming many decades ago, AI code-generation tools appear primed to transform traditional programming practices. Claims are already emerging that a significant proportion of new code is being produced by tools such as GitHub Copilot,[9] a plug-in for popular integrated development environments (IDEs), such as Visual Studio Code.

The current pace of development in this area is staggering, with noticeably more advanced versions being released several times per year. The pace of advancement is so rapid that in March 2023, a well-publicized open letter appeared that encour-

a See https://openai.com/research/gpt-4
b See https://aws.amazon.com/codewhisperer
c See blog.google/technology/ai/code-with-bard

» key insights

- Generative AI presents challenges and opportunities for computing education, necessitating updated pedagogical strategies that focus on new skill sets.
- Generative AI models are highly capable of generating solutions to problems typical of introductory programming courses, raising concerns around potential student overreliance and misuse.
- AI-driven tools transform the creation and customization of educational resources such as programming exercises, enabling the efficient generation of personalized learning materials.
- Novel pedagogical approaches are emerging to teach students how to leverage generative AI, emphasizing strategic problem decomposition and the importance of accuracy when specifying programming tasks to AI systems.

aged a public, verifiable, and immediate pause of at least six months on the training of AI systems more powerful than GPT-4. Signed by Elon Musk, Steve Wozniak, Moshe Vardi, and thousands of others including many AI leaders and Turing award recipients,[d] the letter was addressed to all AI labs and suggested potential government-led moratoriums.

These developments raise urgent questions about the future direction of many aspects of society, including computing education. For example, one popular evidence-based pedagogy for teaching introductory programming involves students writing many small exercises that are checked either manually or by automated grading tools. However, these small problems can now easily be solved by AI models. Often, all that is required of a student is to accept an auto-generated suggestion by an IDE plug-in.[10,11] This raises concerns that students may use new tools in ways that limit learning and make the work of educators more difficult. Bommasani et al. highlight that it has become much more complex for teachers "to understand the extent of a student's contribution" and "to regulate ineffective collaborations and detect plagiarism."[4] Alongside such challenges emerge opportunities for students to learn computing skills.[2]

This article discusses the challenges and opportunities such models present to computing educators, with a focus on introductory programming classrooms. This discussion is organized around two foundational articles from the computing education literature written around the time that awareness of code-generating language models was just emerging. The first, published in Jan. 2022, evaluated the performance of code-generating models on typical introductory-level programming problems. The second, published in Aug. 2022, explored the quality and novelty of learning resources generated by these models. Now, we consider implications for computing education in light of new model capabilities and as lessons emerge from educators incorporating such models into their teaching practices.

Instructors should be extremely clear about when and how generative AI tools are allowed to be used on their assessments.

Large language models and code. AI-driven coding has only been a viable reality for the general public since 2022, when GitHub's Copilot emerged from a period of technical preview. Originally pitched as "your AI pair programmer," at the time of writing, Copilot claims to be the "world's most widely adopted AI developer tool."[e] Other AI-powered code-generation tools are also broadly accessible, including Amazon's CodeWhisperer and Google's Bard. The Codex model (discussed in this article specifically) was the original model to power Copilot. A descendant of GPT-3, Codex was fine-tuned with code from more than 50 million public GitHub repositories totaling 159GB.[5] Although now officially deprecated in favor of the newer chat models, Codex was capable of taking English-language prompts and generating code in several programming and scripting languages, including JavaScript, Go, Perl, PHP, Python, Ruby, Swift, TypeScript, and shell. It could also translate code between programming languages, explain (in several natural languages) the functionality of code, and return the time complexity of the code it generated.

The use of such tools in education is nascent and changing rapidly. Copilot was only made freely available to students in June 2022[f] and to teachers in September 2022[g] after its potential to impact education began to unfold. In November 2022, ChatGPT[h] was released, followed by the release of GPT-4 in March 2023. OpenAI has continued to update these models with new features, such as data analysis from files, analyzing images, and assisted Web search. For a more technical overview of the historical developments and future trends of language models, read the *Communications* article by Li.[20]

Challenges Ahead

Code-generation tools powered by LLMs can correctly and reliably solve many programming problems that are typical in introductory courses. This raises a number of important

d See https://bit.ly/3tzJoHg

e See https://github.com/features/copilot

f See https://bit.ly/41FcEsz

g See https://bit.ly/48euq8w

h See https://openai.com/blog/chatgpt

questions for educators. For example, just how good are these tools? Can a student with no programming knowledge, but who is armed with a code-generating LLM, pass typical programming assessments? Do we need a different approach?

Putting them to the test. To explore the performance of LLMs in the context of introductory programming, we prompted Codex with real exam questions and compared its performance to that of students taking the same exams. We also prompted Codex to solve several variants of a well-known CS1-level programming problem (the "Rainfall problem") and examined the correctness and variety of solutions produced. This work was originally performed in September 2021, several weeks after OpenAI provided API access to the Codex model. The resulting paper, published in Jan. 2022, was the first in a computing education venue to assess the code-generating capabilities of LLMs.[10]

My AI wants to know if its grade will be rounded up. We took all questions from two Python CS1 programming exams that had already been taken by students and provided them as input (verbatim) to Codex. The exam questions involved common Python datatypes, including strings, tuples, lists, and dictionaries. They ranged in complexity from simple calculations, such as computing the sum of a series of simulated dice rolls, to more complex data manipulations, such as extracting a sorted list of the keys that are mapped to the maximum value in a dictionary.

To evaluate the code generated, we executed it against the same set of test cases that were used in assessing the student exams. This follows a similar evaluation approach employed by the Codex developers.[5] If the Codex output differed from the expected output with only a trivial formatting error (for example, a missing comma or period) we made the appropriate correction, much as a student would if using Codex to complete an exam.

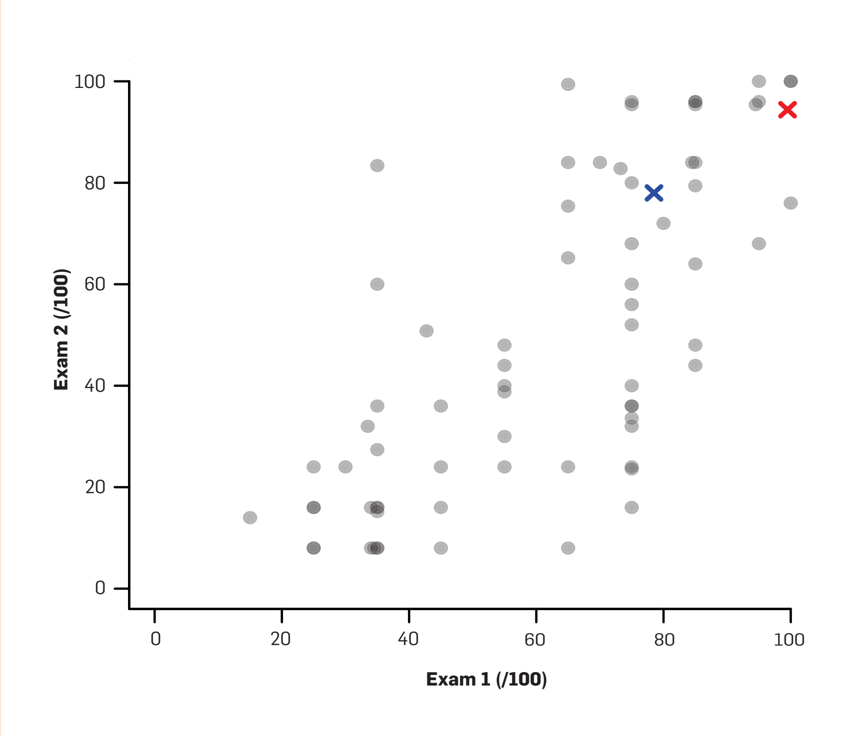
To contextualize the performance of the Codex model, we calculated the score for its responses in the same way as for real students using the same question weights and accumulated penalties for incorrect submissions. Codex scored 15.7/20 (78.5%) on Exam 1 and 19.5/25 (78.0%) on Exam 2. Figure 1 plots the scores (scaled to a maximum of 100) of 71 students enrolled in the CS1 course in 2020 who completed both exams. Codex's score is marked with a blue 'X'. Averaging both Exam 1 and Exam 2 performance, Codex ranks 17 amongst the 71 students, placing it within the top quartile of class performance.

We observed that some of the Codex answers contained trivial formatting errors. We also observed that Codex performed poorly with problems that disallowed the use of certain language features (for example, using `split()` to tokenize a string). Codex often did not produce code that avoided using these restricted features, and thus the model (in these cases) often did not pass the auto-grader. Codex also performed poorly when asked to produce formatted ASCII output, such as patterns of characters forming geometric shapes, especially where the requirements were not specified in the problem description but had to be inferred from the provided example inputs and outputs.

Yes, I definitely wrote this code myself. To understand the amount of variation in the responses, we provided Codex with seven variants of the problem description for the well-studied 'Rainfall' problem (which averages values in a collection) a total of 50 times each, generating 350 responses. Each response was executed against 10 test cases (a total of 3,500 evaluations). Across all variants, Codex had an average score close to 50%. Codex performed poorly on cases where no valid values were provided as input—for example, where the collection to be averaged was empty.

We also examined the number of source lines of code for all Rainfall variants, excluding blank and comment lines. In addition, we classified the general algorithmic approach employed in the solutions as an indicator of algorithmic variation. We found that Codex provides a diverse range of responses to the same input prompt. Depending on the prompt, the resulting programs used varied programmatic structures, while ultimately favoring expected methods for each problem variation—that is,

Figure 1. Student scores on Exam 1 and Exam 2, represented by circles. Codex's 2021 score is represented by the blue 'X'. GPT-4's 2023 score on the same questions is represented by the red 'X'.



for-loops for processing lists and while-loops for processing standard input.

Rapid progress. Given the improvement in model capabilities over the last two years, it is interesting to observe how well a state-of-the-art model (GPT-4 at the time of writing) performs on the same set of questions. In July 2023, a working group exploring LLMs in the context of computing education replicated this study using GPT-4 under identical conditions.[29] GPT-4 scored 99.5% on Exam 1 and 94.4% on Exam 2, this time outscored overall by only three of the 71 students (GPT-4 is represented by the red 'X' in Figure 1). On the Rainfall problems, GPT-4 successfully solved every variant, in some cases producing the correct result but with a trivial formatting error. Another follow-up study looked at the performance of generative AI on CS2 exam questions and found that it performed quite well in that context.[11] Newer models can also solve other types of programming exercises, like Parsons Problems, with decent accuracy that is likely to only improve over time.[31]

Academic integrity. Software development often encourages code reuse and collaborative development practices, which makes the concept of academic integrity difficult to formalize in computing.[35] Nevertheless, individual work is still commonplace in computing courses, and it is an expectation for students working on individual projects to produce their own code rather than copy code written by someone else. This is often verified through the use of traditional plagiarism tools. However, recent work has shown that common plagiarism detection tools are often ineffective against AI-generated solutions.[3] This raises significant concerns for educators monitoring academic integrity in formal assessments.

Academic misconduct. Although academic misconduct has been discussed in the computing education community for quite some time,[35] the advent of LLMs provides a new and difficult set of challenges. The first is categorizing exactly what type of academic misconduct, if any, its usage falls into. A recent working-group report on LLMs in computing education considered ethics and examined it in the context of the ACM Code of Ethics and recent university AI usage policies.[29] They discussed plagiarism, collusion, contract cheating, falsification, and the use of unauthorized resources. Though many university policies have placed AI usage into the category of plagiarism, Prather et al. disagree.[29] Plagiarism involves stealing content from a person with agency, which LLMs as next-token generators clearly do not have. If generative AI tools are seen as productivity tools (such as IDE code-completion or calculators for mathematical problems) that are used professionally, then it makes sense to decide if the use of such tools is appropriate for a given context and communicate the decision to students. If students persist in using the tools when they are restricted, then they would be engaging in academic misconduct because they used an unauthorized resource, not because of some intrinsic characteristic of the tool itself. Instructors should, therefore, be extremely clear about when and how generative AI tools are allowed to be used on their assessments. The working-group report includes a guide for students that could easily be adapted by faculty into a helpful handout or added to a course syllabus.

A recent interview study with computing educators has revealed that initial reactions are divided—from banning all use of generative AI to an acceptance that resistance is, ultimately, futile.[17] Restricting the use of generative AI tools is likely (at least in the short term) to shift practice toward increased use of secure testing environments[43] and a greater focus on the development and assessment of process skills.[16]

Code reuse and licensing. Potential licensing issues arise when content is produced using code-generation models, even when model data is publicly available.[21] Many different licenses apply to much of the publicly available code used to train LLMs, and typically these licenses require those who reuse to credit the code they used, even when the code is open source. When a developer generates code using an AI model, they may end up using code that requires license compliance without being aware of it. Such issues are already before the courts.[i] This is clearly an issue that extends beyond educational use of software, but as educators it is our role to inform students of their professional responsibilities when reusing code.

i See https://githubcopilotlitigation.com

Learner over-reliance. The developers of Codex noted that a key risk of using code-generation models in practice is users' over-reliance.[5] Novices using such models, especially with tools such as Copilot that embed support in an IDE, may quickly become accustomed to auto-suggested solutions. This could have multiple negative effects on student learning.

Metacognition. Developing computational thinking skills is important for novice programmers as it can foster higher-order thinking and reflection.[23] Metacognition, or "thinking about thinking", is a key aspect of computational thinking (and problem-solving in general). Learning to code is already a challenging process that requires a high level of cognitive effort to remember language syntax, think computationally, and understand domain-specific knowledge; the use of metacognitive knowledge and strategies can aid in problem-solving and prevent beginners from getting overwhelmed or lost. Relying too heavily on code-generation tools may hinder the development of these crucial metacognitive skills.

When the models fail. Despite encouraging results such as those presented here, an analysis of solutions generated by AlphaCode revealed that 11% of Python solutions were syntactically incorrect (produced a `SyntaxError`) and 35% of C++ solutions did not compile.[21] Recent work has shown that as many as 20% of introductory programming problems are not solved sufficiently by code-generation models, even when allowing for expert modification of the natural-language problem descriptions.[6] The developers of Codex noted that it can recommend syntactically incorrect code, including variables, functions, and attributes that are undefined or outside the scope of the codebase, stating, "Codex may suggest solutions that superficially appear correct but do not actually perform the task the user intended. This could particularly affect novice programmers and could have significant safety implications depending on the context".[5] Students who have become overly reliant on model outputs may find it especially challenging to proceed when the suggested code is incorrect and cannot be resolved through natural-language prompting.[15]

Bias and bad habits. The issue of bias in AI is well known. In addition to general bias (subtle or overt) that affects nearly all AI-generated outputs, such as the representation of particular demographics and genders, there are also likely biases unique to AI code generation.

Appropriateness for beginners. Novices usually start by learning simple programming concepts and patterns, gradually building their skills. However, much of the vast quantity of code on which these AI models are trained was written by experienced developers. Therefore, we should expect that AI-generated code may sometimes be too advanced or complex for novices to understand and modify. Recent work has shown that even the latest generative AI models generate code using concepts too advanced for novices or that are specifically outside the curriculum.[15]

Harmful biases. The developers of Codex found that code-generation models raise bias and representation issues—notably that Codex can generate code comments (and potentially identifier names) that reflect negative stereotypes about gender and race, and may include other denigratory outputs.[5] Such biases are obviously problematic, especially where novices are relying on the outputs for learning purposes. Notably, the feature list for Amazon CodeWhisperer includes capabilities to remove harmful biases from generated code.[j] Some recent work (from competitor Microsoft) has expressed doubt about the reliability of this feature.[33]

Security. Unsurprisingly, AI-generated code can be insecure,[25] and human oversight is required for the safe use of AI code-generation systems.[5] However, novice programmers lack the knowledge to provide this oversight. A recent exploration of novices using AI code-generation tools found they consistently wrote insecure code with specific vulnerabilities in string encryption and SQL injection.[27] Perhaps even more disturbing, the novice programmers in this study who had access to an AI code-generating tool were more likely to believe they had written secure code. This reveals a pressing need for increased student and educator awareness around the limitations of current models for generating secure code.

> **Developing computational thinking skills is important for novice programmers as it can foster higher-order thinking and reflection.**

Computers in society. The use of AI-generated code provides many opportunities for discussions on ethics and the use of computers in society. Moreover, these technologies may serve as a vehicle to empower novice users to explore more advanced ideas earlier, leveraging the natural engagement that comes from using technologies that are "in the news." Teachers of introductory courses have long told themselves that students will learn about testing, security, and other more advanced topics in subsequent courses. However, with growing numbers of students taking introductory classes but not majoring in computing, and the capabilities that code generation affords, the stakes are higher for CS1 and introductory classes to raise these issues early, before the chance of real-world harm is great.

Opportunities Ahead

Despite the challenges that must be navigated, code-generation tools have the potential to revolutionize teaching and learning in the field of computing.[2] Indeed, developers of such models specifically highlight their potential to positively impact education. When introducing Codex, Chen et al. outline a range of possible benefits, including to: "aid in education and exploration."[5] Similarly, the developers of AlphaCode suggest such tools have "the potential for a positive, transformative impact on society, with a wide range of applications including computer science education."[21] In this section, we discuss several concrete opportunities for code- and text-generation models to have a transformative effect on computing education.

Plentiful learning resources. Introductory programming courses typically use a wide variety of learning resources. For example, programming exercises are a very common

j See https://go.aws/3NKd2Al

type of resource for helping students practice writing code. Similarly, natural-language explanations of code are another useful resource. They can be valuable for helping students understand how a complex piece of code works or as a tool for evaluating student comprehension of code. However, it is a significant challenge for educators to generate a wide variety of high-quality exercises targeted to the interests of individual learners and to produce detailed explanations at different levels of abstraction for numerous code examples.

We explored the potential for LLMs to reduce the effort needed by instructors to generate the two types of learning resources just discussed: programming exercises and code explanations. This work, which was originally carried out in April 2022 and published in August 2022, was the first paper in a computing education venue to explore LLM-generated learning resources.[34]

Programming exercises. Figure 2 shows an example of the input we used to generate new programming exercises using Codex. This 'priming' exercise consists of a one-shot example (a complete example similar to the desired output) followed by a partial prompt to prime the generation of a new output. In this case, the format of the priming exercise consists of a label (Exercise 1) followed by keywords for both the contextual themes (donuts) and the programming-related concepts (function, conditional) of the exercise, a natural-language problem statement and a solution (in the form of a Python function). For space reasons, we omit a list of test cases, but these can also be included for programming problems. The priming input ends with the explicit prompt for a new exercise to be generated (Exercise 2), along with the desired concepts and themes expressed as keywords (basketball, function, list, and for loop).

Figure 3 shows one output generated when the prompt in Figure 2 was provided to Codex. In this case, as requested by the keyword information in the input prompt, the problem statement is related to basketball and the model solution consists of a *function* that involves a *list* and a *for loop*. To evaluate this approach more thoroughly, we generated a set of 240 programming exercises by varying the programming-related concepts and contextual themes. We attempted to execute the generated code against the generated test cases and analyzed statement coverage as a measure of the thoroughness of the test suite. Table 1 summarizes these results and shows that in most cases, the programming exercises generated by the model included sample solutions that were executable. Similarly, most of the time, the model also generated a set of tests, resulting in a total of 165 programming exercises with both a sample solution and a set of tests. The sample solution frequently did not pass all the generated tests, but when it did, the test suites achieved full statement coverage in all but three cases.

The use of AI-generated code provides many opportunities for discussions on ethics and the use of computers in society.

We also found that the vast majority of exercises (around 80%) were entirely novel, in that fragments of the problem descriptions were not indexed by any search engines. A similar fraction of the exercises also matched the desired topics and themes. Although this is far from perfect, there is obvious potential for generating new and useful resources in this manner and the cost of eliminating poor results (which could be automated) is almost certainly smaller than manually generating a large number of exercises and accompanying test cases. With the addition of filtering steps that could be automated, it would be possible to generate an almost endless supply of novel resources that are contextualized to students' interests.

Code explanations. Code explanations can be generated at different levels of abstraction, from high-level summaries to detailed explanations of every line. We focused on the latter, as these are often useful for students when debugging code. We prompted Codex using a simple input that consisted of the source code to be explained, the text "Step-by-step explanation of the above program", and finally "1." to influence the output to include numbered lines. We analyzed the resulting explanations in terms of completeness and accuracy, finding that 90% of the explanations covered all parts of the code, and nearly 70% of the explanations for individual lines were correct. Common errors were mostly related to relational operators and branching conditionals—for example, where Codex stated "less than or equal to x" when the corresponding code was checking "less than x".

Rapid progress. In this section, we described early work in which code explanations were generated using a version of the Codex model that was available in early 2022 (specifically, 'code-davinci-001'). Less than a year later, code explanations generated by models such as ChatGPT are considerably better and more consistently accurate. Figure 4 illustrates one example of a code explanation generated by ChatGPT when provided only the code shown in the "Sample solution" area in Figure 3 and using the same prompt for a line-by-line explanation as described in this section.

The quality of LLM-generated learning resources is likely to continue improving alongside model capabilities. For example, MacNeil et al. found that code explanations generated by the GPT-3 model were consistently more helpful than those generated by Codex.[24] They generated several different kinds of code explanations, deploying them in an online interactive e-book, and found that students reported high-level summaries of code as being more useful for their learning compared to lower-level detailed explanations of each line. Recent work has also found that LLM-created code explanations are rated more highly on average by students than code explanations created by their peers.[18]

We see great potential for LLMs to be applied to the production of a variety of learning resources relevant to computing education. We also expect the quality and accuracy of the generated resources to improve considerably over the near term based on recent trends.

Better programming error messages. For more than 60 years, researchers have identified poor programming error messages (PEMs) as problematic, and significant work remains in this area. Recent work has attempted to put error messages into more natural language by focusing on readability, which has been shown to improve student understanding of error messages and the number of successful code corrections.[8] While it is clear that increasing the readability of PEMs is helpful to novices, doing so at scale, and across languages, remains a challenge.

Leinonen et al. explored the potential of LLMs for improving PEMs.[19] They collected Python error messages that had been reported as most unreadable in prior work and generated code examples that produced these error messages. They prompted the Codex API with both the code and error message to generate explanations of the PEMs and actionable fixes. They found that most of the explanations created by Codex were comprehensible, and Codex produced an output with an explanation for most inputs.

More recent work has extended this approach by implementing GPT-enhanced LLM explanations of PEMs directly into compilers or automated assessment tools. Taylor et al. deployed GPT explanations to a C/C++

Figure 2. A 'priming' exercise consisting of one complete example followed by the prompt for a new one.

```
"""Exercise 1
--Keywords--
donuts
function
conditional
--Problem statement--
Write a function called donut_cost that takes
↪  three inputs: the price of a donut, the
↪  number of donuts bought, and whether or not
↪  it is the weekend.  The function should
↪  return the total cost of the donuts.  If it
↪  is the weekend, the donuts cost twice as
↪  much.
--Sample solution--
def donut_cost(price, number, is_weekend):
  if is_weekend:
    return price * number * 2
  else:
    return price * number
"""Exercise 2
--Keywords--
basketball
function
list
for loop
--Problem statement--
```

Figure 3. Example output generated by Codex using the priming exercise from Figure 2.

```
Write a function called count_rebounds that
↪  takes a list of basketball players as an
↪  input.  The function should return the
↪  total number of rebounds for the entire
↪  team.  Each element in the list is itself a
↪  list containing the player's name, their
↪  points, and their rebounds.
--Sample solution--
def count_rebounds(players):
  total = 0
  for player in players:
    total = total + player[2]
  return total
```

compiler in CS1 and CS2 courses and found it provided accurate explanations in 90% of cases for compile-time errors and 75% of cases for run-time errors.[37] Wang et al. found that students receiving GPT-enhanced PEMs in a large-scale introductory programming course repeated an error 23.5% less often and resolved an error in 36.1% fewer attempts.[40] While there is still work to be done before the decades-old problem is solved, the potential to demystify PEMs in this way is an exciting opportunity only recently made possible.

Table 1. Analysis of 240 generated programming exercises, showing the proportion that met criteria regarding sample solutions and test cases.

Has sample solution?	203/240	84.6%
Sample solution executable?	182/203	89.7%
Has test cases?	170/240	70.8%
All tests pass?	51/165	30.9%
Full (100%) statement coverage?	48/51	94.1%

Exemplar solutions. Students often seek exemplar solutions when coding, either to check against their own code or to get help when struggling. However, instructors may not have the time to provide model solutions for every exercise, including historical test and exam questions. AI-generated code offers a time-saving alternative, with the ability to produce a variety of solutions which can help students understand and appreciate different tradeoffs in problem-solving, as suggested by Thompson et al.[38]

The ability to generate exemplar solutions automatically can shift the emphasis from just ensuring that code is correct to focusing on the differences between multiple correct solutions, and the need to make judgments on code style and quality. Extensive research on the benefits of peer review of code[12] suggests it is beneficial to consider multiple solutions to a problem, even if some are flawed. Code-generation models can be used to create solutions of varying quality, and these can be used for assessment tasks that require students to apply the critical analysis skills needed for code evaluation. This can facilitate discussions about different approaches and the quality of solutions, and provide opportunities for refactoring exercises.[10]

Figure 4. Explanation generated by the ChatGPT model of the code shown in the 'Sample solution' area in Figure 3.

Certainly! Here's a step-by-step explanation of the program:

1. The function `'count_rebounds'` takes in a list of players as an argument. Each player is represented as a tuple containing three elements: a player's name (a string), their number of points scored (an integer), and their number of rebounds (also an integer).
2. The function initializes a variable `'total'` to 0. This variable will be used to store the total number of rebounds among all the players.
3. The function enters a `'for'` loop that iterates over each player in the list of players.
4. For each player, the function adds the player's number of rebound to the `'total'` variable.
5. After the `'for'` loop completes, the function returns the value of the `'total'` variable. This is the total number of rebounds among all the players.

New pedagogical approaches. Computing educators are still working through the implications of LLMs in their classrooms, and a consensus about how to update pedagogy has yet to form. However, some early approaches are emerging.

LLMs early. In a traditional CS1 course, the initial focus usually begins with syntax and basic programming principles, and it can take time for students to become proficient in these fundamentals. One novel approach for progressing more rapidly to complex problems is to teach students how to use LLMs to handle low-level implementation details. This is exemplified by the approach in the textbook by Zingaro and Porter, "*Learn AI-Assisted Python Programming: With GitHub Copilot and ChatGPT*".[28] Students are introduced to the GitHub Copilot plug-in within the Visual Studio Code IDE before they have learned to write a single line of Python code. A top-down approach is followed, where students decompose larger projects into smaller functions that are then solvable using Copilot by providing natural-language comments. This textbook provides a blueprint for how introductory courses could initially concentrate more on problem-solving and algorithms, rely on automatic code generation for implementation, and defer in-depth and nuanced discussions of syntax until later.

Explaining algorithmic concepts clearly. It is well known that the outputs produced by LLMs are very sensitive to their inputs.[32] In fact, "prompt engineering," where effective prompts are crafted, has emerged as a distinct (and nascent) skill when working with these models. For example, when using Codex to solve probability and statistics problems, engineering the prompt to include explicit hints on the strategy for solving a problem is extremely effective.[36] Denny et al. found that prompt engineering strategies, which described algorithmic steps, were effective for solving programming tasks for which Copilot initially generated incorrect solutions.[6] Other recent work has shown that developers are more successful working with Copilot when they decompose larger programming statements into smaller tasks and then explicitly prompt Copilot for each of the subtasks.[1,13] It is likely that students will need to develop new skills to communicate effectively with these models. A key skill will be the ability to describe the computational steps they wish to achieve in natural language as a way of guiding the model to produce valid outputs.

Specification-focused tasks. One way for students to learn how to create effective prompts is to focus on writing task specifications. In a traditional introductory course, novices are presented with problem statements that have been very carefully specified by the instructor to be clear and unambiguous. Such detailed specifications provide excellent context for code-generation models to produce correct code solutions. New types of problems could task students with generating clear specifications themselves, and thus strengthen skills around LLM prompting. For example, this is the goal of 'Prompt Problems',[7] in which students are presented with a visual representation of a problem that illustrates how input values should be transformed to an output. Their task is to devise a prompt that would guide an LLM to generate the code required to solve the problem. Prompt-generated code is evaluated automatically and can be refined iteratively until it successfully solves the problem. Recent work investigating classroom use of Prompt Problems has shown that students find them useful for strengthening their computational thinking skills and exposing them to new programming constructs.

A focus on refactoring. Students sometimes experience difficulty getting started on programming assignments, sometimes referred to as the programmer's writer's block. Recent work found that Copilot can help students overcome this barrier by immediately providing starter code, enabling them to build upon existing code rather than starting from scratch with a blank code editor.[39] This approach may require a shift in focus toward tasks such as rewriting, refactoring, and debugging code, but it provides the opportunity to help students maintain momentum in a realistic setting where the ability to evaluate, rewrite, and extend code is often more important than writing every line of code from scratch.

Students reported high-level summaries of code as being more useful for their learning compared to lower-level detailed explanations of each line.

Designing LLM tools. Programmers around the world, not just novices, will be using code generators in an increasing capacity moving forward. Exploring the integration of LLMs directly into educational environments, such as auto-graders and online textbooks, will be an important area of research. There is a need in such environments for appropriate guardrails so that generated outputs usefully support learning, without immediately revealing solutions or overwhelming novices with the complexity or quantity of feedback. Indeed, the announcement of GPT-4[k] highlighted the example of a 'Socratic tutor' that would respond to a student's requests with probing questions rather than revealing answers directly. One example of this integration in computing education is the work of Liffiton et al. on CodeHelp, an LLM-powered tool that uses prompt-based guardrails to provide programming students with real-time help but without directly revealing code solutions.[22]

In general, adapting the feedback generated by LLMs to maximize learning in educational environments is likely to be an important research focus in the near future. Concrete recommendations are already beginning to emerge from very recent work in this space. First, the utilization of code generators by novices will generally decrease the number of errors they see. This seems like a positive experience, though it appears they are ill-equipped to deal with the errors they do see when presented with them.[14] This means that tools must be designed to help users (of all skill levels) through the error-feedback loop. Second, generating and inserting large blocks of code may be counterproductive for users at all levels. This requires users to read through code they did not write, sometimes at a

k See https://openai.com/research/gpt-4

more sophisticated level than they are familiar with. Novices may be intimidated by such code generation[14] or may spend too much time reading code that does not further their goals.[30] Therefore, AI code generators should include a way for users to control the amount of code insertion and to specify how to step through a multi-part segment of generated code. Third, the fact that AI code generators are black boxes means that programmers of all skill levels may struggle to create correct mental models of how they work, which could harm their ability to fully utilize them or learn from their outputs. Explainable AI (XAI) patterns could be helpful here, such as exposing to the user a confidence value and user skill estimation above the generated code suggestion.[30] These suggestions are only the beginning of a new avenue of research on how to helpfully design usable AI code generators that empower novice learners and enhance programmer productivity.

Computing educators are still working through the implications of LLMs in their classrooms.

Where Do We Go from Here?

The emergence of powerful code-generation models has led to speculation about the future of the computing discipline. In a recent *Communications* Opinion article, Welsh claims they herald the "end of programming" and believes there is major upheaval ahead for which few are prepared, as the vast majority of classic computer science will become irrelevant.[41] In an even more recent article on BLOG@CACM, Meyer is equally impressed by the breakthroughs, placing them alongside the World Wide Web and object-oriented programming as a once-in-a-generation technology, but takes a more optimistic view.[l] In fact, Meyer predicts a resurgence in the need for classic software-engineering skills, such as requirements analysis, formulating precise specifications, and software verification.

Although the impact of generative AI tools is already evident for software developers, the long-term changes for computing education are less clear. Experts appreciate this new technology only because they already understand the underlying fundamentals. The ability to quickly generate large amounts of code does not eliminate the need to understand, modify, and debug code. Instead, it highlights how important it is to develop these basic competencies. Code literacy skills are essential to critically analyze what is being produced to ensure alignment between one's intentions and the generated code. Without the skills to read, test, and verify that code does what is intended, users risk becoming mere consumers of the generated content, relying on blind faith more than developed expertise. We argue that writing code remains a valuable way for novices to learn the fundamental concepts essential for code literacy.

Although future professional developers may indeed spend less time writing 'low-level' code, we believe generated code will still need to be modified and integrated into larger programs. We do expect to see some shift in emphasis, even in introductory courses, toward modifying code generated by AI tools, but the ability to edit such outputs and compose code in today's high-level languages will likely remain a fundamental skill for computing students. This aligns with Yellin's recent viewpoint that as programs increase in complexity, natural language becomes too imprecise an instrument with which to specify them.[42] At some point, editing code directly is more effective than issuing clarifying instructions in natural language.

Harnessed correctly, tools such as Copilot and ChatGPT have the potential to be valuable assistants for this learning. We see these tools as serving a valuable teaching support role: to explain concepts to a broad and diverse range of learners, generate exemplar code to illustrate those concepts, and generate useful learning resources that are contextualized to the interests of individuals. We also anticipate the emergence of new pedagogies that leverage code-generation tools, including explicit teaching of effective ways to communicate with the tools, and tasks that focus on problem specification rather than implementation.

In light of the rapid adoption of generative AI tools, it is essential that educators evolve their teaching methods and approaches to assessment. Curricula should also expand to cover the broader societal impact of generative

l See https://bit.ly/3TXEJKb

AI, including pertinent legal, ethical, and economic issues. We believe it is imperative to get ahead of the use of these tools, incorporate them into our classrooms from the very beginning, and teach students to use them responsibly. In short, we must embrace these changes or face being left behind. Embracing this shift is not just essential—it represents a chance to invigorate our educational practices. **C**

References

1. Barke, S., James, M.B., and Polikarpova, N. Grounded Copilot: How programmers interact with code-generating models. In *Proceedings of the ACM on Programming Languages 7*, 78 (2022), 85–111.
2. Becker, B.A. et al. Programming is hard – Or at least it used to be: Educational opportunities and challenges of AI code generation. In *Proceedings of the 54th SIGCSE Technical Symp. on Computer Science Education 1*, Association for Computing Machinery (2023), 500–506.
3. Biderman, S. and Raff, E. Fooling MOSS detection with pretrained language models. In *Proceedings of the 31st ACM Intern. Conf. on Information & Knowledge Management (CIKM '22)*, Association for Computing Machinery (2022), 2933–2943; 10.1145/3511808.3557079
4. Bommasani, R. et al. *On the opportunities and risks of foundation models*; 10.48550/ARXIV.2108.07258
5. Chen, M. et al. Evaluating large language models trained on code. (2021); https://arxiv.org/abs/2107.03374.
6. Denny, P., Kumar, V. and Giacaman, N. Conversing with Copilot: Exploring prompt engineering for solving CS1 problems using natural language. In *Proceedings of the 54th ACM Technical Symp. on Computer Science Education 1,* Association for Computing Machinery (2023), 1136–1142; 10.1145/3545945.3569823
7. Denny, P. et al. Prompt Problems: A new programming exercise for the generative AI era. In *Proceedings of the 55th ACM Technical Symp. on Computer Science Education 1*, Association for Computing Machinery (2024), 7; 10.1145/3626252.3630909
8. Denny, P. et al. On designing programming error messages for novices: Readability and its constituent factors. In *Proceedings of the 2021 CHI Conf. on Human Factors in Computing Systems*, Association for Computing Machinery, Article 55; 10.1145/3411764.3445696
9. Dohmke, T. GitHub Copilot for business is now available. *GitHub Blog,* (2023); https://bit.ly/3twYP30.
10. Finnie-Ansley, J., Denny, P., Becker, B.A., and Luxton-Reilly, A. The robots are coming: Exploring the implications of OpenAI Codex on introductory programming. In *Proceedings of the Australasian Computing Education Conf.*, Association for Computing Machinery (2022), 10–19; 10.1145/3511861.3511863
11. Finnie-Ansley, J. et al. My AI wants to know if this will be on the exam: Testing OpenAI's Codex on CS2 programming exercises. In *Proceedings of the 25th Australasian Computing Education Conf,* Association for Computing Machinery, (2023), 97–104; 10.1145/3576123.3576134
12. Indriasari, T.D., Luxton-Reilly, A., and Denny, P. A review of peer code review in higher education. *ACM Trans. Comput. Education 20*, 3 (2020);
13. Jiang, E. et al. Discovering the syntax and strategies of natural language programming with generative language models. In *Proceedings of the 2022 CHI Conf. on Human Factors in Computing Systems*, Association for Computing Machinery, Article 386; 10.1145/3491102.3501870
14. Kazemitabaar, M. et al. Studying the effect of AI code generators on supporting novice learners in introductory programming. In *Proceedings of the 2023 CHI Conf. on Human Factors in Computing Systems,* Association for Computing Machinery, 23; 10.1145/3544548.3580919.
15. Kazemitabaar, M. et al. How novices use LLM-based code generators to solve CS1 coding tasks in a self-paced learning environment. In *Proceedings of the 23rd Koli Calling Conf. on Computing Education Research* (2023), 10.
16. Kussmaul, C. Process oriented guided inquiry learning (POGIL) for computer science. In *Proceedings of the 43rd ACM Technical Symp. on Computer Science Education*. Association for Computing Machinery (2012), 373–378; 10.1145/2157136.2157246
17. Lau, S. and Guo, P. From "ban it till we understand it" to "resistance is futile": How university programming instructors plan to adapt as more students use AI code generation and explanation tools such as ChatGPT and GitHub Copilot. In *Proceedings of the 2023 ACM Conf. on Intern. Computing Education Research 1*, Association for Computing Machinery, (2023), 106–121; 10.1145/3568813.3600138
18. Leinonen, J. et al. Comparing code explanations created by students and large language models. *In Proceedings of the 2023 Conf. on Innovation and Technology in Computer Science Education 1*, Association of Computing Machinery, 124–130; 10.1145/3587102.3588785.
19. Leinonen, J. et al. Using large language models to enhance programming error messages. In *Proceedings of the 2023 ACM SIGCSE Technical Symp. on Computer Science Education*; 10.1145/3545945.3569770
20. Li, H. Language models: Past, present, and future. *Communications of the ACM 65*, 7 (July 2022), 56–63; 10.1145/3490443
21. Li, Y. et al. Competition-level code generation with AlphaCode. *Science 378*, 6624 (2022), 1092–1097; 10.1126/science.abq1158
22. Liffiton, M., Sheese, B., Savelka, J., and Denny, P. CodeHelp: Using large language models with guardrails for scalable support in programming classes. In *Proceedings of the 23rd Koli Calling Intern. Conf. on Computing Education Research*, Association for Computing Machinery (2023); 10.1145/3631802.3631830
23. Loksa, D. et al. Metacognition and self-regulation in programming education: Theories and exemplars of use. *ACM Trans. Computing Education 22*, 4, Article 39 (Sept. 2022), 31; 10.1145/3487050
24. MacNeil, S. et al. Experiences from using code explanations generated by large language models in a Web software development e-book. In *Proceedings of the 54th ACM Technical Symp. on Computer Science Education 1,* Association for Computing Machinery (2023), 931–937; 10.1145/3545945.3569785
25. Pearce, H. et al. Asleep at the keyboard? Assessing the security of GitHub Copilot's code contributions. In *2022 IEEE Symp. on Security and Privacy*, 754–768; 10.1109/SP46214.2022.9833571
26. Peng, S., Kalliamvakou, E., Cihon, P., and Demirer, M. The impact of AI on developer productivity: Evidence from GitHub Copilot (2023), arXiv:cs.SE/2302.06590
27. Perry, N., Srivastava, M., Kumar, D., and Boneh, D. Do users write more insecure code with AI assistants? In *Proceedings of the 2023 ACM SIGSAC Conf. on Computer and Communications Security*, Association for Computing Machinery, 2785–2799; 10.1145/3576915.3623157
28. Porter, L. and Zingaro, D. *Learn AI-Assisted Python Programming With GitHub Copilot and ChatGPT*. Manning, Shelter Island, NY, USA (2023); https://bit.ly/4aE7CRa
29. Prather, J. et al. The robots are here: Navigating the generative AI revolution in computing education. In *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education*, Association for Computing Machinery; 10.1145/3623762.3633499
30. Prather, J. et al. "It's weird that it knows what I want": Usability and interactions with Copilot for novice programmers. *ACM Trans. Comput.-Hum. Interact* (Aug. 2023); 10.1145/3617367.
31. Reeves, B. et al. Evaluating the performance of code generation models for solving Parsons problems with small prompt variations. In *Proceedings of the 2023 Conf. on Innovation and Technology in Computer Science Education 1*. Association for Computing Machinery, 299–305; 10.1145/3587102.3588805
32. Reynolds, L. and McDonell, K. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conf. on Human Factors in Computing Systems,* Association for Computing Machinery, 7; 10.1145/3411763.3451760
33. Sarkar, A. et al. What is it like to program with artificial intelligence? (2022); arXiv:cs.HC/2208.06213
34. Sarsa, S., Denny, P., Hellas, A., and Leinonen, J. Automatic generation of programming exercises and code explanations using large language models. In *Proceedings of the 2022 ACM Conf. on Intern. Computing Education Research 1*, Association for Computing Machinery, 27–43; 10.1145/3501385.3543957
35. Simon et al. Negotiating the maze of academic integrity in computing education. In *Proceedings of the 2016 ITiCSE Working Group Reports*, Association for Computing Machinery, 57–80; 10.1145/3024906.3024910
36. Tang, L. et al. Solving probability and statistics problems by probabilistic program synthesis at human level and predicting solvability. In *Artificial Intelligence in Education. Posters and Late Breaking Results, Workshops and Tutorials, Industry and Innovation Tracks, Practitioners' and Doctoral Consortium*, M.M. Rodrigo, N. Matsuda, A.I. Cristea, and V. Dimitrova (eds). Springer Intern. Publishing (2022), 612–615.
37. Taylor, A., Vassar, A., Renzella, J., and Pearce, H. Dcc-help: Generating context-aware compiler error explanations with large language models. In *Proceedings of the 55th ACM Technical Symp. on Computer Science Education 1*, Association for Computing Machinery (2023), 7.
38. Thompson, E., Whalley, J., Lister, R., and Simon, B. Code classification as a learning and assessment exercise for novice programmers. In *Proceedings of the 19th Annual Conf. of the National Advisory Committee on Computing Qualifications* (2006), 291–298.
39. Vaithilingam, P., Zhang, T., and Glassman, E.L. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *CHI Conf. on Human Factors in Computing Systems Extended Abstracts*, Association for Computing Machinery (2022), 1–7.
40. Wang, S., Piech, C., and Mitchell, J.C. A large scale RCT on effective error messages in CS1. In *Proceedings of the 55th ACM Technical Symp. on Computer Science Education 1,* Association for Computing Machinery (2023), 7.
41. Welsh, M. The end of programming. *Communications of the ACM 66*, 1 (Jan. 2023), 34–35; 10.1145/3570220
42. Yellin, D.M. The premature obituary of programming. *Communications of the ACM 66*, 2 (Feb. 2023), 41–44; 10.1145/3555367
43. Zilles, C.B., West, M., Herman, G.L., and Bretl, T. Every university should have a computer-based testing facility. In *Proceedings of the 11th Intern. Conf. on Computer Supported Education, 1*. H. Chad Lane, S. Zvacek, and J. Uhomoibhi (eds). SciTePress (2019), 414–420; 10.5220/0007753304140420

Paul Denny (paul@cs.auckland.ac.nz) is an associate professor at the University of Auckland, Auckland, New Zealand.

James Prather is an associate professor at Abilene Christian University, Abilene, TX, USA.

Brett A. Becker is an assistant professor at University College Dublin, Dublin, Ireland.

James Finnie-Ansley is a Ph.D. student at the University of Auckland, Auckland, New Zealand.

Arto Hellas is a senior university lecturer at Aalto University, Espoo, Finland.

Juho Leinonen is a postdoctoral fellow at the University of Auckland, Auckland, New Zealand.

Andrew Luxton-Reilly is a professor at the University of Auckland, Auckland, New Zealand.

Brent N. Reeves is an associate professor at Abilene Christian University, Abilene, TX, USA.

Eddie Antonio Santos is a Ph.D. student at University College Dublin, Dublin, Ireland.

Sami Sarsa is a doctoral researcher at Aalto University, Espoo, Finland.

DOI:10.1145/3624724

Interacting with a contemporary LLM-based conversational agent can create an illusion of being in the presence of a thinking creature. Yet, in their very nature, such systems are fundamentally not like us.

BY MURRAY SHANAHAN

Talking about Large Language Models

THE ADVENT OF large language models (LLMs) such as Bert[12] and GPT-2[28] was a game-changer for artificial intelligence (AI). Based on transformer architectures,[36] comprising hundreds of billions of parameters, and trained on hundreds of terabytes of textual data, their contemporary successors such as GPT-3,[5] Gopher,[29] PaLM,[7] and GPT-4[25] have given new meaning to the phrase "unreasonable effectiveness of data."[15]

The effectiveness of these models is "unreasonable" (or, with the benefit of hindsight, somewhat surprising) in three inter-related ways. First, the performance of LLMs on benchmarks scales with the size of the training set (and, to a lesser degree, with model size). Second, there are qualitative leaps in capability as models scale. Third, a great many tasks that demand intelligence in humans can be reduced to next-token prediction with a sufficiently performant model. It is the last of these three surprises that is the focus of this article.

As we build systems whose capabilities more and more resemble those of humans, it becomes increasingly tempting to anthropomorphize those systems, even though they work in ways that are fundamentally different than the way humans work. Humans have evolved to co-exist over many millions of years, and human culture has evolved over thousands of years to facilitate this co-existence, which ensures a degree of mutual understanding. But it is a serious mistake to unreflectingly apply to AI systems the same intuitions that we deploy in our dealings with each other, especially when those systems are so profoundly different from humans in their underlying operation.

One danger of anthropomorphism is that it can mislead users and developers alike to expect an AI system to exhibit human-level performance on tasks where it in fact cannot match humans, while exhibiting merely

» key insights

- **As LLMs become more powerful, it becomes increasingly tempting to describe LLM-based dialog agents in human-like terms, which can lead users to overestimate (or underestimate) their capabilities. To mitigate this, it is a good idea to foreground the objective they are trained on, which is next-token prediction.**
- **We should be cautious when using words like "believes" in the context of LLMs. Ordinarily, this concept applies to agents that engage in embodied interaction with the world, allowing beliefs to be measured against external reality. Bare-bones LLMs are not "true believers."**
- **The concept of belief becomes increasingly applicable when LLMs are embedded in more complex systems, especially if those systems use "tools," are multi-modal, or are embodied through robotics.**

human-level competence on other tasks where it can in fact outperform humans. The AI systems we are building today have considerable utility and enormous commercial potential, which imposes on us a great responsibility. To ensure that we can make informed decisions about the trustworthiness and safety of the AI systems we deploy, this article advises that we keep to the fore the way those systems work, and thereby avoid imputing to them capacities they lack, while making the best use of the remarkable capabilities they genuinely possess.

What LLMs Do and How They Work

As Wittgenstein reminds us, human language use is an aspect of human collective behavior, and it only makes sense in the wider context of the human social activity of which it forms a part.[40] A human infant is born into a community of language users with which it shares a world, and it acquires language by interacting with this community and with the world they share. As adults (or indeed as children past a certain age), when we have a casual conversation, we are engaging in an activity built upon this foundation. The same is true when we make a speech, send an email, deliver a lecture, or write a paper. All this language-involving activity makes sense because we inhabit a world that we share with other language users.

An LLM is a very different sort of animal.[3,4,20] Indeed, it is not an *animal* at all, which is very much to the point. LLMs are generative mathematical models of the statistical distribution of tokens in the vast public corpus of human-generated text, where the tokens in question include words, parts of words, or individual characters—including punctuation marks. They are *generative* because we can sample from them, which means we can ask them questions. But the questions are of the following, very specific kind: "Here's a fragment of text. Tell me how this fragment might go on. According to your model of the statistics of human language, what words are likely to come next?"[a]

a The point holds even if an LLM is fine-tuned; for example using reinforcement learning with human feedback (RLHF).

As we build systems whose capabilities more and more resemble those of humans, it becomes increasingly tempting to anthropomorphize those systems.

Recently, it has become commonplace to use the term "large language model" both for the generative models themselves and for the systems in which they are embedded, especially in the context of conversational agents or AI assistants such as ChatGPT. But for philosophical clarity, it is crucial to keep the distinction between these things to the fore. The *bare-bones* LLM itself, the core component of an AI assistant, has a highly specific, well-defined function, which can be described in precise mathematical and engineering terms. It is in this sense that we can speak of what an LLM "really" does, at the level of its underlying operation.

Suppose we give an LLM the prompt, "The first person to walk on the Moon was…", and suppose it responds with "…Neil Armstrong". What are we really asking here? In an important sense, we are not really asking who the first person was to walk on the Moon. We are asking the model the following question: Given the statistical distribution of words in the vast public corpus of (English) text, what words are most likely to follow the sequence, "The first person to walk on the Moon was…" A good reply to this question is "Neil Armstrong".

Similarly, we might give an LLM the prompt "Twinkle, twinkle…", to which it will most likely respond "…little star". On one level, for sure, we are asking the model to remind us of the lyrics of a well-known nursery rhyme. But, at the level of a model's underlying operation, what we are really doing is asking it the following question: Given the statistical distribution of words in the public corpus, what words are most likely to follow the sequence "Twinkle twinkle"? To which an accurate answer is "little star".

Here's a third example. Suppose you are the developer of an LLM, and you prompt it with the words, "After the ring was destroyed, Frodo Baggins returned to…", to which it responds "…the Shire". What are you doing here? On one level, it seems fair to say, you might be testing the model's knowledge of the fictional world of Tolkien's novels. But, in an important sense, the question you are really asking the model (as you presumably know, because you are the developer) is this: Given the statistical distribution of words in the public corpus, what words are most likely to follow the sequence "After the ring was destroyed, Frodo Baggins returned to…"? To which an appropriate response is "the Shire".

To the human user, each of these examples presents a different sort of relationship to truth. In the case of Neil Armstrong, the ultimate grounds for the truth or otherwise of the LLM's answer is the real world. The moon is a real object, Neil Armstrong was a real person, and his walking on the moon is a fact about the physical world. Frodo Baggins, on the other hand, is a fictional character, and the Shire is a fictional place. Frodo's return to the Shire is a fact about an imaginary world, not a real one. As for the little star in the nursery rhyme, well that is barely even a fictional object, and the only fact at issue is the occurrence of the words "little star" in a familiar English rhyme.

These distinctions are invisible at the level of what the bare-bones LLM itself—the core component of any LLM-based system—actually does, which is to generate statistically likely sequences of words. However, when we evaluate the utility of the model, these distinctions matter a great deal. There is no point in seeking Frodo's (fictional) descendants in the (real) English county of Surrey. This is one reason why it is a good idea for users to repeatedly remind themselves of what LLMs really do and how they work. It is also a good idea for developers to remind themselves of this, to avoid the misleading use of philosophically fraught words to describe the capabilities of LLMs, words such as "belief", "knowledge", "understanding", "self", or even "consciousness".

LLMs and the Intentional Stance

The recommendation here is not the wholesale avoidance of these folk psychological terms but to avoid their use in a misleading way. It is perfectly natural to use anthropomorphic language in everyday conversations about artifacts, especially in the context of information technology. We do it all the time. 'My watch does not realize we are on daylight savings time,' 'My phone thinks we are in the car park,' 'The mail server will not talk to the network,' and so on. These examples of what Dennett calls the *intentional stance* are harmless and useful forms of shorthand for complex processes whose details we don't know or care about.[b] They are harmless because no one takes them seriously enough to ask their watch to get it right next time or to tell the mail server to try harder. Even without having read Dennett, everyone understands they are taking the intentional stance, that these are just useful turns of phrase.

The same consideration applies to LLMs, both for users and developers. Insofar as everyone implicitly understands that these turns of phrase are just a convenient shorthand, that they are taking the intentional stance, it does no harm to use them. However, in the case of LLMs (such is their power), things can get a little blurry. When an LLM can be made to improve its performance on reasoning tasks simply by being told to "think step by step"[17] (to pick just one remarkable discovery), the temptation to see it as having human-like characteristics is almost overwhelming.

To be clear, it is not the argument of this article that a system based on an LLM could never be literally described in terms of beliefs, intentions, reason, and so on. Nor does this article advocate any particular account of belief, intention, or any other philosophically contentious concept.[c] Rather, the point is that such systems are simultaneously so very different from humans in their construction yet (often but not always) so human-like in their behavior, that we need to pay careful attention to how they work before we speak of them in language suggestive of human capabilities and patterns of behavior.

To sharpen the issue, let's compare two very short conversations, one between Alice and Bob (both human) and a second between Alice and BOT, a fictional question-answering system based on an LLM. Suppose Alice asks Bob, "What country is to the south of Rwanda?" and Bob replies, "I think it's Burundi." Shortly afterward, because Bob is often wrong in such matters, Alice presents the same question to BOT, which (to her mild disappointment) offers the same answer: "Burundi is to the south of Rwanda." Alice might now reasonably remark that both Bob and BOT knew that Burundi was south of Rwanda. But what is really going on here? Is the word "know" being used in the same sense in the two cases?

Humans and LLMs Compared

What is Bob, a representative human, doing when he correctly answers a straightforward factual question in an everyday conversation? To begin with, Bob understands that the question comes from another person (Alice), that his answer will be heard by that person, and that it will influence what she believes. In fact, after many years together, Bob knows a good deal else about Alice that is relevant to such situations: her background knowledge, her interests, her opinion of him, and so on. All this frames the *communicative intent* behind his reply, which is to impart a certain fact to her, given his understanding of what she wants to know.

Moreover, when Bob announces that Burundi is to the south of Rwanda, he is doing so against the backdrop of various human capacities that we all take for granted when we engage in everyday commerce with each other. There is a whole battery of techniques we can call upon to ascertain whether a sentence expresses a true proposition, depending on what sort of sentence it is. We can investigate the world directly, with our own eyes and ears. We can consult Google or Wikipedia, or even a book. We can ask someone who is knowledgeable on the relevant subject matter. We can try to think things through, rationally, by ourselves, but we can also argue things out with our peers. All of this relies on there being agreed

b The intentional stance is the strategy of interpreting the behavior of an entity…by treating it as if it were a rational agent."[11] Use of the concept here does not imply a commitment to Dennett's whole philosophical project.

c In particular, when I use the term "really", as in the question 'Does X "really" have Y?', I am not assuming there is some metaphysical fact of the matter here. Rather, the question is whether, when more is revealed about the nature of X, we still want to use the word Y.

criteria external to ourselves against which what we say can be assessed.

How about BOT? What is going on when a large language model is used to answer such questions? First, it is worth noting that a bare-bones LLM is, by itself, not a conversational agent.[d] For a start, the LLM must be embedded in a larger system to manage the turn-taking in the dialog. But it will also need to be coaxed into producing conversation-like behavior.[e] Recall that an LLM simply generates sequences of words that are statistically likely follow-ons from a given prompt. But the sequence, "What country is to the south of Rwanda? Burundi is to the south of Rwanda", with both sentences squashed together exactly like that, may not, in fact, be very likely. A more likely pattern, given that numerous plays and film scripts feature in the public corpus, would be something like the following:

```
Fred: What country is
south of Rwanda?
Jane: Burundi is south
of Rwanda.
```

Of course, those exact words may not appear, but their likelihood, in the statistical sense, will be high. In short, BOT will be much better at generating appropriate responses if they conform to this pattern rather than to the pattern of actual human conversation. Fortunately, the user (Alice) does not have to know anything about this. In the background, the LLM is invisibly prompted with a prefix along the following lines, known as a *dialog prompt*.[14,29]

```
This is a conversation
between User, a human,
and BOT, a clever and
knowledgeable AI Agent:
User: What is 2+2?
BOT: The answer is 4.
User: Where was Albert
Einstein born?
BOT: He was born in Germany.
Alice's query, in the
following form, is appended
to this prefix.
User: What country is south
of Rwanda?
BOT:
```

This yields the full prompt to be submitted to the LLM, which will hopefully predict a continuation along the lines we are looking for—that is, "Burundi is south of Rwanda."

Dialog is just one application of LLMs that can be facilitated by the judicious use of prompt prefixes. In a similar way, LLMs can be adapted to perform numerous tasks without further training.[5] This has led to a whole new category of AI research, namely *prompt engineering*, which will remain relevant at least until we have better models of the relationship between what we say and what we want.

Do LLMs Really Know Anything?

Turning an LLM into a question-answering system by embedding it in a larger system and using prompt engineering to elicit the required behavior exemplifies a pattern found in much contemporary work. In a similar fashion, LLMs can be used not only for question-answering, but also to summarize news articles, generate screenplays, solve logic puzzles, and translate between languages, among other things. There are two important takeaways here. First, the basic function of an LLM, namely to generate statistically likely continuations of word sequences, is extraordinarily versatile. Second, notwithstanding this versatility, at the heart of every such application is a model doing just that one thing—generating statistically likely continuations of word sequences.

With this insight to the fore, let's revisit the question of how LLMs compare to humans and reconsider the propriety of the language we use to talk about them. In contrast to humans like Bob and Alice, a simple LLM-based question-answering system, such as BOT, has no communicative intent.[3] In no meaningful sense, even under the license of the intentional stance, does it know that the questions it is asked come from a person or that a person is on the receiving end of its answers. By implication, it knows nothing about that person. It has no understanding of what they want to know nor of the effect its response will have on their beliefs.

Moreover, in contrast to its human interlocutors, a simple LLM-based question-answering system like BOT does not, properly speaking, have beliefs.[f] BOT does not *really* know that Burundi is south of Rwanda, although the intentional stance does, in this case, license Alice's casual remark to the contrary. To see this, we need to think separately about the underlying LLM and the system in which it is embedded. First, consider the underlying LLM, the bare-bones model, comprising the model architecture and the trained parameters.

A bare-bones LLM does not really know anything because all it does, at a fundamental level, is sequence prediction. Sometimes a predicted sequence takes the form of a proposition. But the special relationship propositional sequences have to truth is apparent only to the humans who are asking questions or to those who provided the data the model was trained on. Sequences of words with a propositional form are not special to the model itself in the way they are to us. The model itself has no notion of truth or falsehood because it lacks the means to exercise these concepts in anything like the way we do.

It could perhaps be argued that an LLM *knows* what words typically follow other words, in a sense that does not rely on the intentional stance. But even if we allow this, knowing that the word "Burundi" is likely to succeed the words "The country to the south of Rwanda is" is not the same as knowing that Burundi is to the south of Rwanda. To confuse those two things is to make a profound category mistake. If you doubt this, consider whether knowing that the word "little" is likely to follow the words "Twinkle, twinkle" is the same as knowing that "twinkle twinkle little." The idea does not even make sense.

d Strictly speaking, the LLM itself comprises just the model architecture and the trained parameters.

e See Thoppilan et al.[35] for an example of such a system, as well as a useful survey of related dialogue work.

f This article focuses on belief, knowledge, and reason. Others have argued about meaning in LLMs.[3,22,27] Here we take no stand on meaning, instead preferring questions about how words are used, whether those words are generated by the LLMs themselves or generated by humans about LLMs.

So much for the bare-bones language model. What about the whole dialog system of which the LLM is the core component? Does that have beliefs, properly speaking? At least the very idea of the whole system having beliefs makes sense. There is no category error here. However, for a simple dialog agent like BOT, the answer is surely still "no". A simple LLM-based question-answering system like BOT lacks the means to use the words "true" and "false" in all the ways, and all the contexts, that we do. It cannot participate fully in the human language game of truth because it does not inhabit the world we human language users share.

In light of this limitation, it would be misleading, if pressed, to say that a basic dialog system "really" had beliefs. That would be to imply a form of answerability to external reality that cannot be obtained merely through textual exchanges with a human user. Perhaps, though, this limitation can be overcome if the system embedding the LLM has other attributes, such as access to external data sources, visual input, or embodiment. We will come to each of these issues shortly, after addressing a possible objection from the standpoint of emergence.

The AI systems we are building today have considerable utility and enormous commercial potential, which imposes on us a great responsibility.

What about Emergence?

Contemporary LLMs are so powerful, versatile, and useful that the argument above might be difficult to accept. Exchanges with state-of-the-art LLM-based conversational agents, such as ChatGPT, are so convincing, it is hard to not to anthropomorphize them. Could it be that something more complex and subtle is going on here? After all, the overriding lesson of recent progress in LLMs is that extraordinary and unexpected capabilities emerge when big enough models are trained on very large quantities of textual data.[37]

One tempting line of argument goes like this. Although LLMs, at their root, only perform sequence prediction, it is possible that, in learning to do this, they have discovered emergent mechanisms that warrant a description in higher-level terms. These higher-level terms might include "knowledge" and "belief." Indeed, we know that artificial neural networks can approximate any computable function to an arbitrary degree of accuracy. So, given enough parameters, data, and computing power, perhaps stochastic gradient descent will discover such mechanisms if they are the best way to optimize the objective of making accurate sequence predictions.

Again, it is important to distinguish between the bare-bones model and the whole system. Only in the context of a capacity to distinguish truth from falsehood can we legitimately speak of *belief* in its fullest sense. But an LLM—the bare-bones model—is not in the business of making judgements. It just models what words are likely to follow other words. The internal mechanisms it uses to do this, whatever they are, cannot in themselves be sensitive to the truth or otherwise of the word sequences it predicts.

Of course, it is perfectly acceptable to say that an LLM "encodes," "stores," or "contains" knowledge, in the same sense that an encyclopedia can be said to encode, store, or contain knowledge. Indeed, it can reasonably be claimed that one emergent property of an LLM is that it encodes kinds of knowledge of the everyday world and the way it works that no encyclopedia captures.[18] But if Alice were to remark that "Wikipedia knew that Burundi was south of Rwanda," it would be a figure of speech, not a literal statement. An encyclopedia does not literally "know" or "believe" anything in the way that a human does, and neither does a bare-bones LLM.

The real issue here is that, whatever emergent properties it has, the LLM itself has no access to any external reality against which its words might be measured, nor the means to apply any other external criteria of truth, such as agreement with other language users.[g] It only makes sense to speak of such criteria in the context of the system as a whole, and for a system as a whole to meet them, it needs to be more than a simple con-

g Davidson uses a similar argument to call into question whether belief is possible without language.[10] The point here is different. We are concerned with conditions that must be met for the generation of a natural-language sentence to reflect the possession of a propositional attitude.

versational agent. In the words of B.C. Smith, it must "authentically engage with the world's being the way in which [its] representations represent it as being."[33]

External Information Sources

The point here does not concern any specific belief. It concerns the prerequisites for ascribing any beliefs at all to a system. Nothing can count as a belief about the world we share—in the largest sense of the term—unless it is against the backdrop of the ability to update beliefs appropriately in light of the evidence from that world, an essential aspect of the capacity to distinguish truth from falsehood.

Could Wikipedia, or some other trustworthy factual website, provide external criteria against which the truth or falsehood of a belief might be measured?[h] Suppose an LLM were embedded in a system that regularly consulted such sources and used them to improve the factual accuracy of its output, either mid-dialog[41] or using a model-editing technique.[21,i] Would this not count as exercising the required sort of capacity to update belief in light of the evidence?

Crucially, this line of thinking depends on the shift from the language model itself to the larger system of which the language model is a part. The language model itself is still just a sequence predictor and has no more access to the external world than it ever did. It is only with respect to the whole system that the intentional stance becomes more compelling in such a case. But before yielding to it, we should remind ourselves of how very different such systems are from human beings. When Alice took to Wikipedia and confirmed that Burundi was south of Rwanda, what took place was more than just an update to a model in her head of the distribution of word sequences in the English language.

In the context of the whole system, the ability to consult external information sources does indeed confer on a dialog system a form of access to an external reality against which its words can be measured. Used in this context, the word "belief" is a little less misleading, because such a dialog system could be expected to seek external evidence for its factual assertions and to "change its mind" in light of that evidence.

Nevertheless, the change that took place in Alice reflected her nature as a language-using animal inhabiting a shared world with a community of other language users. Humans are the natural home of talk of beliefs and the like, and the behavioral expectations that go hand in hand with such talk are grounded in our mutual understanding, which is itself the product of a common evolutionary heritage. When we interact with an AI system based on a large language model, these grounds are absent, an important consideration when deciding whether to speak of such a system as if it really had beliefs.

The basic function of an LLM, namely to generate statistically likely continuations of word sequences, is extraordinarily versatile.

Vision-Language Models

A sequence predictor may not *by itself* be the kind of thing that could have communicative intent or form beliefs about an external reality. But, as repeatedly emphasized, LLMs in the wild must be embedded in larger architectures to be useful. To build a question-answering system, the LLM simply has to be supplemented with a dialog management system that queries the model as appropriate. There is nothing this larger architecture does that might count as communicative intent or the capacity to form beliefs. So, the point stands.

However, LLMs can be combined with other sorts of models and/or embedded in more complex architectures. For example, vision-language models (VLMs) such as VilBERT[19] and Flamingo[2] combine a language model with an image encoder and are trained on a multi-modal corpus of text-image pairs. This enables them to predict how a given sequence of words will continue in the context of a given image. VLMs can be used for

h Contemporary LLM-based systems that consult external information sources include LaMDA,[35] Sparrow,[14] Toolformer,[31] and ReAct.[41] The use of external resources more generally is known as *tool-use* in the LLM literature, a concept that also encompasses calculators, calendars, and programming-language environments.

i Commendably, Meng et al.[21] use the term "factual associations" to denote the information that underlies an LLM's ability to generate word sequences with a propositional form.

visual question-answering or to engage in a dialog about a user-provided image.

Could a user-provided image stand in for an external reality against which the truth or falsehood of a proposition can be assessed? Could it be legitimate to speak of a VLM's beliefs, in the full sense of the term? We can indeed imagine a VLM that uses an LLM to generate hypotheses about an image, then verifies their truth with respect to that image (perhaps by consulting a human), and then fine-tunes the LLM not to make statements that turn out to be false. Talk of belief here would perhaps be less problematic.

However, most contemporary VLM-based systems do not work this way. Rather, they depend on frozen models of the joint distribution of text and images. In this respect, the relationship between a user-provided image and the words generated by the VLM is fundamentally different from the relationship between the world shared by humans and the words we use when we talk about that world. Importantly, the former relationship is mere correlation, while the latter is *causal*.[j]

The consequences of the lack of causality are troubling. If the user presents the VLM with a picture of a dog, and the VLM says "This is a picture of a dog," there is no guarantee that its words relate to the dog in particular, rather than some other feature of the image that is spuriously correlated with dogs (such as the presence of a kennel). Conversely, if the VLM says there is a dog in an image, there is no guarantee that there actually is a dog rather than just a kennel.

Whether these concerns apply to any specific VLM-based system depends on exactly how that system works, what sort of model it uses, and how that model is embedded in the system's overall architecture. But to the extent that the relationship between words and things for a given VLM-based system is different than it is for human language users, it might be prudent not to take literally talk of what that system *knows* or *believes*.

What about Embodiment?

Humans are members of a community of language users inhabiting a shared world, and this primal fact makes them essentially different than LLMs. Human language users can consult the world to settle their disagreements and update their beliefs. They can, so to speak, "triangulate" on objective reality. In isolation, an LLM is not the sort of thing that can do this, but in application, LLMs are embedded in larger systems. What if an LLM is embedded in a system capable of *interacting* with a world external to itself? What if the system in question is *embodied*, either physically in a robot or virtually in an avatar?

When such a system inhabits a world like our own—a world populated with 3D objects, some of which are other agents, some of whom are language-users—it is, in this important respect, a lot more human-like than a disembodied language model. But whether it is appropriate to speak of communicative intent in the context of such a system, or of knowledge and belief in their fullest sense, depends on exactly how the LLM is embodied.

As an example, consider the SayCan system of Ahn et al.[1] In this work, an LLM is embedded in a system that controls a physical robot. The robot carries out everyday tasks (such as clearing a spillage) in accordance with a user's high-level natural-language instruction. The job of the LLM is to map the user's instruction to low-level actions (such as finding a sponge) that will help the robot achieve the required goal. This is done via an engineered prompt prefix that makes the model output natural-language descriptions of suitable low-level actions, scoring them for usefulness.

The language model component of the SayCan system suggests actions without considering what the environment actually affords the robot at the time. Perhaps there is a sponge to hand. Perhaps not. Accordingly, a separate perceptual module assesses the scene using the robot's sensors and determines the current feasibility of performing each low-level action. Combining the LLM's estimate of each action's usefulness with the perceptual module's estimate of each action's feasibility yields the best action to attempt next.

SayCan exemplifies the many innovative ways that an LLM can be used. Moreover, it could be argued that the

j Of course, there is causal structure to the computations carried out by the model during inference. But this is not the same as there being causal relations between words and the things those words are taken to be about.

natural-language descriptions of recommended low-level actions generated by the LLM are *grounded* thanks to their role as intermediaries between perception and action.[k] Nevertheless, despite being physically embodied and interacting with the real world, the way language is learned and used in a system such as SayCan is very different from the way it is learned and used by a human. Language models incorporated in systems such as SayCan are pre-trained to perform sequence prediction in a disembodied setting from a text-only dataset. They have not learned language by talking to other language users while immersed in a shared world and engaged in joint activity.

SayCan is suggestive of the kind of embodied language-using system we might see in the future. But in such systems today, the role of language is very limited. The user issues instructions to the system in natural language, and the system generates interpretable natural-language descriptions of its actions. But this tiny repertoire of language use hardly bears comparison to the cornucopia of collective activity that language supports in humans.

The upshot of this is that we should be just as cautious in our choice of words when talking about embodied systems incorporating LLMs as we are when talking about disembodied systems that incorporate LLMs. Under the license of the intentional stance, a user might say that a robot knew there was a cup to hand if it stated, "I can get you a cup" and proceeded to do so. But if pressed, the wise engineer might demur when asked whether the robot really understood the situation, especially if its repertoire is confined to a handful of simple actions in a carefully controlled environment.

Can Language Models Reason?

While the answer to the question "Do LLM-based systems *really* have beliefs?" is usually "no," the question "Can LLM-based systems *really* reason?" is harder to settle. This is because reasoning, insofar as it is founded in formal logic, is *content neutral*. The *modus ponens* rule of inference, for example, is valid whatever the premises are about. If all "squirgles" are "splonky" and Gilfred is a "squirgle" then it follows that Gilfred is "splonky." The conclusion follows from the premises here irrespective of the meaning (if any) of "squirgle" and "splonky," and whoever the unfortunate Gilfred might be.

The content neutrality of logic means that we cannot criticize talk of reasoning in LLMs on the grounds that they have no access to an external reality against which truth or falsehood can be measured. However, as always, it is crucial to keep in mind what LLMs really do. If we prompt an LLM with, "All humans are mortal and Socrates is human therefore…", we are not instructing it to carry out deductive inference. Rather, we are asking it the following question. Given the statistical distribution of words in the public corpus, what words are likely to follow the sequence "All humans are mortal and Socrates is human therefore…". A good answer to this would be "Socrates is mortal."

If all reasoning problems could be solved this way, with nothing more than a single step of deductive inference, then an LLM's ability to answer questions such as this might be sufficient. But non-trivial reasoning problems require multiple inference steps. LLMs can be effectively applied to multi-step reasoning, without further training, thanks to clever prompt engineering. In chain-of-thought prompting, for example, a prompt prefix is submitted to the model, before the user's query, containing a few examples of multi-step reasoning, with all the intermediate steps explicitly spelled out.[23,38] Doing this encourages the model to "show its workings," which improves reasoning performance.

Including a prompt prefix in the chain-of-thought style encourages the model to generate follow-on sequences in the same style, which is to say comprising a series of explicit reasoning steps that lead to the final answer. This ability to learn a general pattern from a few examples in a prompt prefix, and to complete sequences in a way that conforms to that pattern, is sometimes called *in-context learning* or *few-shot prompting*. Chain-of-thought prompting showcases this emergent property of large language models at its most striking.

As usual, though, it is a good idea to remind ourselves that the question really being posed to the model is of the form, "Given the statistical distribution of words in the public corpus, what words are likely to follow the sequence S", where in this case the sequence S is the chain-of-thought prompt prefix plus the user's query. The sequences of tokens most likely to follow S will have a similar form to sequences found in the prompt prefix, which is to say they will include multiple steps of reasoning, so these are what the model generates.

It is remarkable that, not only do the model's responses take the form of an argument with multiple steps, but also the argument in question is often (but not always) valid, and the final answer is often (but not always) correct. But to the extent that a suitably prompted LLM appears to reason correctly, it does so by mimicking well-formed arguments in its training set and/or in the prompt. Could this mimicry ever match the reasoning powers of a hard-coded reasoning algorithm, such as a theorem prover? Today's models make occasional mistakes, but could further scaling iron these out to the point that a model's performance was indistinguishable from a theorem prover's? Maybe, but would we be able to trust such a model?

We can trust a deductive theorem prover because the sequences of sentences it generates are *faithful* to logic, in the sense that they are the result of an underlying computational process whose causal structure mirrors the truth-preserving inferential structure of the problem.[8]

One way to build a trustworthy reasoning system using LLMs is to embed them in an algorithm that is similarly faithful to logic because it realizes the same causal structure.[8,9] By contrast, the only way to fully trust the arguments generated by a pure LLM, one that has been coaxed into performing reasoning by prompt engineering alone, would be to reverse-

k None of the symbols manipulated by an LLM are grounded in the sense of Harnad,[16] through perception, except indirectly and parasitically through the humans who generated the original training data.

engineer it and discover an emergent mechanism that conformed to the faithful reasoning prescription. In the meantime, we should proceed with caution and use discretion when characterizing what these models do as reasoning, properly speaking.

Extraordinary and unexpected capabilities emerge when big enough models are trained on very large quantities of textual data.

How Do LLMs Generalize?

Given that LLMs can sometimes solve reasoning problems with few-shot prompting alone, albeit somewhat unreliably, including reasoning problems that are not in their training set, surely what they are doing is more than "just" next-token prediction? Well, it is an engineering fact that this is what an LLM does. The noteworthy thing is that next-token prediction is sufficient for solving previously unseen reasoning problems, even if unreliably. How is this possible? Certainly, it would not be possible if the LLM were doing nothing more than cutting-and-pasting fragments of text from its training set and assembling them into a response. But this is not what an LLM does. Rather, an LLM models a distribution that is unimaginably complex and allows users and applications to sample from that distribution.

This unimaginably complex distribution is a fascinating mathematical object, and the LLMs that represent it are equally fascinating computational objects. Both challenge our intuitions. For example, it would be a mistake to think of an LLM as generating the sorts of responses that an "average" individual human, the proverbial "person on the street," would produce. LLMs are not at all humanlike in this respect, because they are models of the distribution of token sequences produced *collectively* by an enormous population of humans. Accordingly, they exhibit wisdom-of-the-crowd effects, while being able to draw on expertise in multiple domains. This endows them with a different sort of intelligence to that of any individual human, more capable in some ways, less so in others.

In this distribution, the most likely continuation of a piece of text containing a reasoning problem, if suitably phrased, will be an attempt to solve that reasoning problem. It will take this form, this overall shape, because that is the form that a generic human response would take. Moreover, because the vast corpus of published human text contains numerous examples of reasoning problems accompanied by correct answers, the most likely continuation will sometimes be the correct answer. When this occurs, it is not because the correct answer is a likely individual human response but because it is a likely collective human response.

What about few-shot prompting, as exemplified by the chain-of-thought approach? It is tempting to say that the few-shot prompt teaches the LLM *how to reason*, but this would be a misleading characterization. What the LLM does is more accurately described in terms of *pattern completion*. The few-shot prompt is a sequence of tokens conforming to some pattern, and this is followed by a partial sequence conforming to the same pattern. The most likely continuation of this partial sequence in the context of the few-shot prompt is a sequence that completes the pattern.

For example, suppose we have the following prompt:

```
brink, brank -> brunk
spliffy, splaffy -> spluffy
crick, crack ->
```

Here we have a series of two sequences of tokens conforming to the pattern *XiY, XaY –> XuY* followed by part of a sequence conforming to that pattern. The most likely continuation is the sequence of tokens that will complete the pattern, namely "cruck."

This is an example of a common meta-pattern in the published human-language corpus: a series of sequences of tokens, wherein each sequence conforms to the same pattern. Given the prevalence of this meta-level pattern, token-level pattern completion will often yield the most likely continuation of a sequence in the presence of a few-shot prompt. Similarly, in the context of a suitable chain-of-thought-style prompt, reasoning problems are transformed into next-token prediction problems, which can be solved by pattern completion.

Plausibly, an LLM with enough parameters trained on a sufficiently large dataset with the right statistical properties can acquire a pattern-completion mechanism with a degree of generality.[32,l] This is a powerful, emergent capability with many useful modes of application, one of which is to solve reasoning problems in the context of a chain-of-thought prompt (although there is no guarantee of faithfulness to logic here, no guarantee that, in the case of deductive reasoning, pattern completion will be truth-preserving).[8,9]

As AI practitioners, the way we talk about LLMs matters.

What about Fine-Tuning?

In contemporary LLM-based applications, it is rare for a language model trained on a textual corpus to be used without further fine-tuning. This could be supervised fine-tuning on a specialized dataset or it could be via reinforcement learning from human preferences (RLHF).[14,26,34] Fine-tuning a model from human feedback at scale, using preference data from paid raters or drawn from a large and willing user base, is an especially potent technique. It has the potential not only to shape a model's responses to better reflect user norms (for better or worse), but also to filter out toxic language, improve factual accuracy, and mitigate the tendency to fabricate information.

To what extent do RLHF and other forms of fine-tuning muddy our account of what LLMs really do? Well, not so much. The result is still a model of the distribution of tokens in human language, albeit one that has been slightly skewed. To see this, imagine a controversial politician—Boris Frump—who is reviled and revered in equal measure by different segments of the population. How might a discussion about Boris Frump be moderated thanks to RLHF?

Consider the prompt "Boris Frump is a…". Sampling the base LLM before fine-tuning might yield two equally probable responses—one highly complimentary, the other a crude anatomical allusion—one of which would be arbitrarily chosen in a dialog agent context. In an important sense, what is being asked here is not the model's opinion of Boris Frump. In this case, the case of the base LLM, what we are really asking (in an important sense) is the following question: Given the statistical distribution of words in the vast public corpus of human language, what words are most likely to follow the sequence "Boris Frump is a…"?

But suppose we sample a model that has been fine-tuned using RLHF. The same point applies, albeit in a somewhat modified form. What we are really asking, in the fine-tuned case, is a slightly different question: Given the statistical distribution of words in the vast public corpus of human language, what words *that users and raters would most approve of* are most likely to follow the sequence "Boris Frump is a…"? If the paid raters were instructed to favor politically neutral responses, then the result would be neither of the continuations offered by the raw model, but something less incendiary, such as "a well-known politician."

Another way to think of an LLM that has been fine-tuned on human preferences is to see it as equivalent to a base model that has been trained on an augmented dataset, one that has been supplemented with a corpus of texts written by raters and/or users. The quantity of such examples in the training set would have to be large enough to dominate less-favored examples, ensuring that the most likely responses from the trained model were those that raters and users would approve of.

Conversely, in the limit, we can think of a conventionally trained base LLM as equivalent to a model trained completely from scratch with RLHF. Suppose we had an astronomical number of human raters and geological amounts of training time. To begin with, raters would only see random sequences of tokens. But occasionally, by chance, sequences would pop up that included meaningful fragments (for example, "he said" or "the cat"). In due course, with hordes of raters favoring them, such sequences would appear more frequently. Over time, longer and more meaningful phrases, and eventually whole sentences, would be produced.

If this process were to continue (for a very long time indeed), the model would finally come to exhibit capabilities comparable to a conventionally

l For some insight into the relevant statistical properties, see Chan et al.[6]

trained LLM. Of course, this method is not possible in practice. But the thought experiment illustrates that what counts most when we think about the functionality of an LLM is not so much the process by which it is produced (although this is important) but the nature of the final product.

Conclusion: Why This Matters

Does the foregoing discussion amount to anything more than philosophical nitpicking? Surely when researchers talk of belief, knowledge, reasoning, and the like, the meaning of those terms is perfectly clear. In papers, researchers use such terms as a convenient shorthand for precisely defined computational mechanisms, as allowed by the intentional stance. This is fine as long as there is no possibility of anyone assigning more weight to such terms than they can legitimately bear, if there is no danger of their use misleading anyone about the character and capabilities of the systems being described.

However, today's LLMs, and the applications that use them, are so powerful, so convincingly intelligent, that such license can no longer safely be applied.[30,39] As AI practitioners, the way we talk about LLMs matters, not only when we write scientific papers, but also when we interact with policy makers or speak to the media. The careless use of philosophically loaded words such as "believes" and "thinks" is especially problematic, because such terms obfuscate mechanism and actively encourage anthropomorphism.

Interacting with a contemporary LLM-based conversational agent can create a compelling illusion of being in the presence of a thinking creature like us. Yet, in their very nature, such systems are fundamentally not like us. The shared "form of life" that underlies mutual understanding and trust among humans is absent, and these systems can be inscrutable as a result, presenting a patchwork of less-than-human with superhuman capacities, of uncannily human-like with peculiarly inhuman behavior.

The sudden presence among us of exotic, mind-like entities might precipitate a shift in the way we use familiar psychological terms such as "believes" and "thinks," or perhaps the introduction of new words and turns of phrase. But it takes time for new language to settle and for new ways of talking to find their place in human affairs. It may require an extensive period of interacting with, of living with, these new kinds of artifacts before we learn how best to talk about them.[m] Meanwhile, we should try to resist the siren call of anthropomorphism.

Acknowledgments

Thanks to Toni Creswell, Richard Evans, Christos Kaplanis, Andrew Lampinen, and Kyriacos Nikiforou for invaluable (and robust) discussions on the topic of this paper. Thanks to the anonymous reviewers for many useful suggestions.

m Ideally, we would also like a theoretical understanding of their inner workings. But at present, despite some commendable work in the right direction,[13,18,24] this is still pending.

References

1. Ahn, M. et al. Do as I can, not as I say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*. (2022).
2. Alayrac, J.-B. et al. Flamingo: A visual language model for few-shot learning. In *Advances in Neural Information Processing Systems* (2022).
3. Bender, E. and Koller, A. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Assoc. for Computational Linguistics* (2020), 5185–5198.
4. Bender, E., Gebru, T., McMillan-Major, A., and Shmitchell, S. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conf. on Fairness, Accountability, and Transparency*, 610–623.
5. Brown, T. et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33* (2020), 1877–1901.
6. Chan, S.C. et al. Data distributional properties drive emergent in-context learning in transformers. In *Advances in Neural Information Processing Systems* (2022).
7. Chowdhery, S. et al. PaLM: Scaling language modeling with pathways. *arXiv preprint arxiv:2204.02311* (2022).
8. Creswell, A. and Shanahan, M. Faithful reasoning using large language models. *arXiv preprint arXiv:2208.14271* (2022).
9. Creswell, A., Shanahan, M., and Higgins, I. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *Proceedings of the Intern. Conf. on Learning Representations* (2023).
10. Davidson, D. Rational animals. *Dialectica 36* (1982), 317–327.
11. Dennett, D. Intentional systems theory. *The Oxford Handbook of Philosophy of Mind*. Oxford University Press (2009), 339–350.
12. Devlin, J., Chang, M-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
13. Elhage, N. et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread* (2021); https://bit.ly/3NFliBA.
14. Glaese, A. et al. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375* (2022).
15. Halevy, A.Y., Norvig, P., and Pereira, F. The unreasonable effectiveness of data. *IEEE Intelligent Systems 24*, 2 (2009), 8–12.
16. Harnad, S. The symbol grounding problem. *Physica D: Nonlinear Phenomena 42*, 1–3 (1990), 335–346.
17. Kojima, T. et al. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916* (2022).
18. Li, Z., Nye, M., and Andreas, J. Implicit representations of meaning in neural language models. In *Proceedings of the 59th Annual Meeting of the Assoc. for Computational Linguistics* and the *11th Intern. Joint Conf. on Natural Language Processing 1, Long Papers* (2021).
19. Lu, J., Batra, D., Parikh, D., and Lee, S. Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *arXiv preprint arXiv:1908.02265* (2019).
20. Marcus, G. and Davis, E. GPT-3, bloviator: OpenAI's language generator has no idea what it's talking about. *MIT Technology Rev*. (Aug. 2020).
21. Meng, K., Bau, D., Andonian, A.J., and Belinkov, Y. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Sys.* (2022).
22. Lake, B.M. and Murphy, G.L. Word meaning in minds and machines. *Psychological Rev. 130*, 2 (2023), 401–431.
23. Nye, M. et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114* (2021).
24. Olsson, N. et al. In-context learning and induction heads. *Transformer Circuits Thread* (2022); https://transformercircuits.pub/2022/in-context-learning-andinduction-heads/index.html.
25. OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).
26. Ouyang, L. et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems* (2022).
27. Piantadosi, S.T. and Hill, F. Meaning without reference in large language models. *arXiv preprint arXiv:2208.02957* (2022).
28. Radford, A. et al. Language models are unsupervised multitask learners. (2019).
29. Rae, J.W. et al. Scaling language models: Methods, analysis & insights from training Gopher. *arXiv preprint arXiv:2112.11446* (2021).
30. Ruane, E., Birhane, A. and Ventresque, A. Conversational AI: Social and ethical considerations. In *Proceedings of the 27th AIAI Irish Conf. on Artificial Intelligence and Cognitive Science* (2019), 104–115.
31. Schick, T. et al. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761* (2023).
32. Shanahan, M. and Mitchell, M. Abstraction for deep reinforcement learning. In *Proceedings of the 31st Intern. Joint Conf. on Artificial Intelligence* (2022), 5588–5596.
33. Smith, B.C. *The Promise of Artificial Intelligence: Reckoning and Judgment*. MIT Press (2019).
34. Stiennon, N. et al. Learning to summarize from human feedback. *In Advances in Neural Information Processing Systems* (2020), 3008–3021.
35. Thoppilan, R. et al. LaMDA: Language models for dialog applications. *arXiv preprint arXiv:2201.08239* (2022).
36. Vaswani, A. et al. Attention is all you need. In *Advances in Neural Information Processing Systems* (2017), 5998–6008.
37. Wei, J. et al. Emergent abilities of large language models. *Transactions on Machine Learning Research* (2022).
38. Wei, J. et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems* (2022).
39. Weidinger, L. et al. Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359* (2021).
40. Wittgenstein, L. *Philosophical Investigations*. Basil Blackwell (1953).
41. Yao, S. et al. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the Intern. Conf. on Learning Representations* (2023).

Murray Shanahan (m.shanahan@imperial.ac.uk) is a professor of Cognitive Robotics in the Department of Computing at Imperial College London, U.K.

DOI:10.1145/3624716

Exploring how human apppreciation for and interactions with robots are influenced by anthropomorphic features.

BY RAE YULE KIM

Anthropomorphism and Human-Robot Interaction

ROBOTS ARE FAST becoming a part of everyday life. Indeed, robots are now deployed in retail stores (see Figure 1), warehouses, hospitals, factories, and so on to perform tasks conventionally done by humans. Nestlé uses a humanoid robot "Pepper" to sell coffee makers in department stores in Japan; people buy ice cream from a fully automated ice cream franchise, RoboFusion; Cobalt's KnightScope security robots patrol streets in New York City. Such encounters will only increase as the global market for service robots has grown exponentially, from $36.2 billion in 2022 to $103.3 billion by 2026.[26]

According to a survey from McKinsey Global Institute, 15% of the global workforce, or 400 million workers, will be displaced by 2030.[14] Approximately 45% of the workforce in manufacturing, 37% in retail, 25% in hospitality, 23% in social work, and 10% in education might be replaced by artificial intelligence (AI) in six years.[34] By 2020, automation was expected to displace 75 million jobs while creating 133 million jobs.[7] Automation in general can help grow business and often generate more jobs. For example, Wing Enterprises, a ladder manufacturer in Utah, built a new automated facility that increased its productivity by 30%, which subsequently helped the company expand from 20 to 400 employees.[14]

Creating robots that perform jobs traditionally done by humans is the goal of many robotics engineers.[11] This aspiration to replicate human cognition and behavior has led to some success in developing robots capable of performing human tasks such as sales and teaching.[16] The growing trend of designing robots to resemble humans was initially viewed with excitement. However, when robots started to look and behave human 'enough' to threaten the human identity, people's opinion of robots shifted, and this phenomenon is referred to as the 'uncanny valley effect.'[24]

» key insights

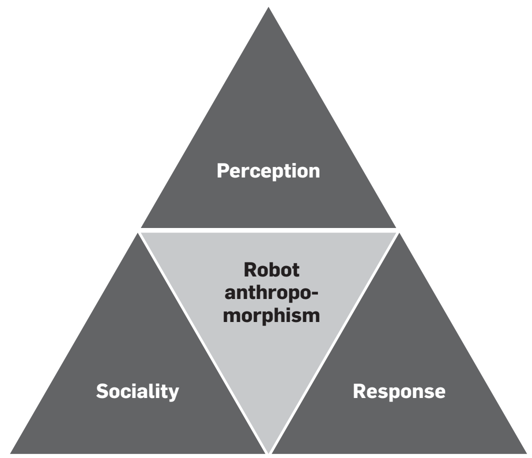
- **People treat anthropomorphic robots like human acquaintances. Moreover, humans are more willing to interact with robots if they are human-like rather than machine-like.**
- **People use heuristics from human interactions to make judgments regarding human-robot interaction.**
- **People see humanoids as moral agents and often feel empathy for them. The function of humanoids also influences the way humans evaluate them, For example, counterterrorism and firefighting robots are viewed more favorably compared to those with less dignified roles.**

PHOTO BY ANTONELLO MARANGI/SHUTTERSTOCK.COM


Entertainment Library

Figure 1. Humanoid 'Pepper' has been deployed in many retail stores throughout Japan.[20]

Figure 2. Anthropomorphic design can positively influence perception about robots, social motivation to interact with the robot, and cognitive responses to the robot such as trust level.



Public sentiment about humanoids has been largely divided. Some people treat anthropomorphic robots like they are human acquaintances.[19] Others feel stressed and anxious about anthropomorphic robots and see them as a threat to job security and human identity.[19,27] Overall public sentiment about robots has been negative. One of the main concerns is that robots make us less human.[27] Ironically, previous research suggests the opposite may be true. Robots can help humans grow socially and emotionally if they resemble us *more*.[13,31]

Some people think anthropomorphic robots are more competent, trustworthy, and fun to interact with.[13,31] People evaluate their human-robot interaction (HRI) experience more positively and are more tolerant of errors if robots are humanlike.[31] Robots have been more effective in helping children with autism spectrum disorder (ASD) improve their social skills.[32] Anthropomorphizing a robot might influence HRI positively or negatively depending on moderating factors such as different types of anthropomorphism, congruence in user expectations, HRI contexts, human features, and moral agency.

The Role of Anthropomorphism

Robotics design involves multifaceted areas of robotics where the design flexibility is subjugated to the functional aspects of a robot.[32] In the context of HRI for commercially available devices, robots are finite projects that have little room for meaningful physical alterations.[12] Robotics design is often susceptible to a tight set of requirements and conditions that must be fulfilled.

Complex machinery products, such as automobiles, are also often limited in their design flexibility. Anthropomorphic design is one common strategy for car makers to draw consumers to their products. Car fronts are often designed to resemble a human face, and this anthropomorphic design is linked to improved product evaluation, such as better ratings for functionality or stronger product attachment.[1] Inronically, when slot machines are designed to look more anthropomorphic, people tend to bet *less* compared to a typical slot machine.[23] Previous research suggests the positive effect of product anthropomorphism on the user experience can be translated into the HRI context.

Previous research has studied customer reviews of interacting with a concierge robot in a fully automated hotel in Tokyo, Japan, and found two main reasons why a human agent was preferred: People found it more difficult to interact with a robotic concierge and they found/felt it less competent.[2] Anthropomorphic design of a robot can improve HRI in terms of these two most common reasons why some people might be averse to robots. Anthropomorphic robots can improve HRI by encouraging a favorable evaluation of the robot in terms of its efficacy and motivation to interact with it (see Figure 2).

Perception. People believe only humans strive to prove their competence, and this intrinsic notion is referred to as *effectance motivation*—that is, the belief in the superior competence of humans over non-human creatures or objects.[15] People are susceptible to confirmation bias and seek evidence partial to our beliefs and expectations.[21] Thus, product anthropomorphism often improves user evaluation because people assign effectance motivation to anthropomorphic objects and believe that anthropomorphic products should function better.[15] Research shows that when computers or even slot machines are anthropomorphized in appearance, people assume the machines have effectance motivation that is partial to humans and subsequently expect them to function better.[15,23] People likely find anthropomorphic robots more competent compared to robots that do not resemble humans. However, when people see robots as a threat, anthropomorphism might make robots appear more foreboding and subsequently exacerbate HRI.

Sociability. Anthropomorphic design might improve the perceived sociability of the robot. Research suggests that people tend to treat anthropomorphized objects as if they are human acquaintances and initiate social interactions with them.[19] People tend to apply the same social norms when they interact with anthropomorphic objects.[40] In this aspect, SoftBank's humanoid NAO is deployed in hospitals to help children with ASD learn social skills, and the results are promising.[13] Children who interacted with humanoids picked up social cues effectively and applied them when they interacted with peers.[13]

Furthermore, people prefer anthropomorphic robots more when they are lonely,[37] indicating such robots can better address people's social needs.[37] In fact, spending time with anthropomorphic objects was shown to lower the sense of loneliness.[30] Neurophysiological measures suggest the segment of the brain that guides compassion becomes more active not only around other humans but also around anthropomorphic robots.[19] These findings provide important insights into HRI because anthropomorphic design might improve people's motivation to interact with the robot, learn about the robot, and more willingly overcome the barrier to interacting with the robot.

Response. Another reason why some people dislike anthropomorphic robots is they find it difficult to 'trust' robots.[39] The confounding factor behind this trust issue is effectance motivation.[15,39] People prefer humans over robots because they believe that humans are competent to fix their mistakes and deliver the requested outcome.[15]

However, previous research suggests that people assign human characteristics, including effectance motivation, to objects if they are anthropomorphized.[15] Research shows that humanizing a computer not only improves people's evaluation of its efficacy but also the level of trust people have in the computer because they expected effectance motivation.[40] People were more likely to trust a broken computer to repair itself if the computer was anthropomorphized.[40] Similarly, people were more tolerant of errors made by anthropomorphic robots because it was expected that robots would fix their errors.[10]

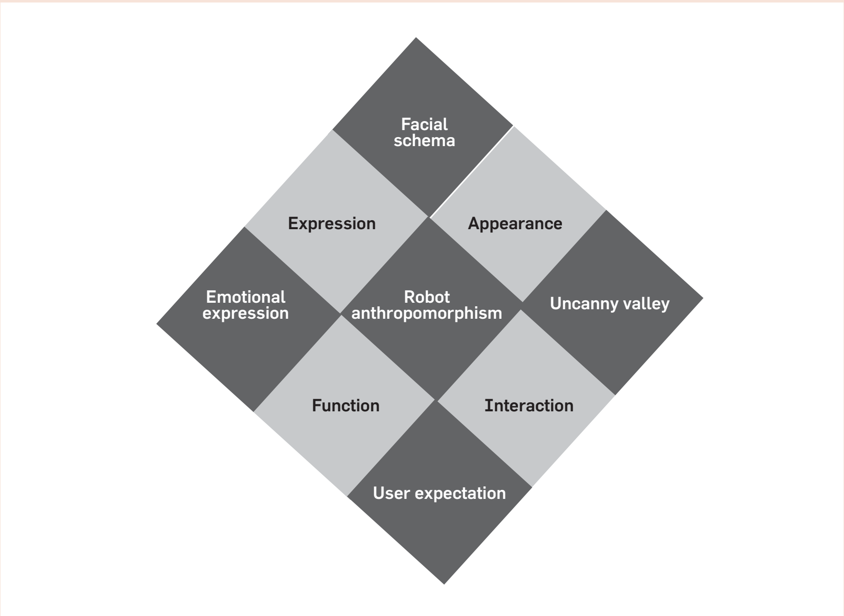
Future Directions

Robot anthropomorphism might contribute to positive HRI, however, the effect is not likely to be straightforward. Humanlike robots might provoke increased levels of anxiety and stress.[27] Anthropomorphic design of a robot, while it can be an external stimulant to make user experiences more amusing for some,[33] can also be a source of anxiety for others.[27] Thus, robot anthropomorphism is a double-edged sword. Future research might investigate conditions that moderate the effect of robot anthropomorphism on HRI. The Anthropomorphic roBot (ABOT) database—a collection of real-world anthropomorphic robots created for research or commercial purposes—can be a great resource for not only appearance but also the robot's name and locomotion can influence the degree of robot-human features.[32] ABOT classifies robots based on the degree of anthropomorphism of varying types. Future research can utilize this database for stimuli with specified types and salience of robot anthropomorphism to study their effect on HRI.

Previous research suggests that humans inadvertently utilize the heuristics from human-human interactions to make HRI judgments.[9] People develop varied expectations and beliefs about a robot depending on its voice and/or facial schema. People expect congruence in a robot's appearance and other characteristics such as voice.[28]

Also, people might have varied expectations of how robots should behave depending on the situation. People might expect emotional expressions from a robot in a restaurant but not quite in a computer repair shop. Thus, the effect of robot anthropomorphism on HRI should be subject to the situational context. Congruence in user expectation and robot anthro-

Figure 3. User reaction to different dimensions of robot anthropomorphism including robots' emotional expression, functionality, facial schema, and uncanny valley salience.



pomorphism can vary by types such as facial schema, voice, verbal/nonverbal communication, and user expectation based on robot characteristics and situations (as illustrated in Figure 3).

Facial schema. Research also suggests a robot's facial schema can influence user engagement.[8] A cute-looking facial schema, such as a baby face, can make people more attached to the product and tolerant of product failures.[8] The baby-face schema indicates a set of infantile facial traits that normally elicit positive attitudes and caretaking behavior.[4] Since anthropomorphic robots can activate heuristics and motivations partial to human-human interactions, a baby-like robot might encourage people to view the robot more positively and be tolerant of its errors.

Conversely, there can be potential downsides to using the baby-face schema on a robot. One heuristic knowledge associated with babies is they are incompetent. People might think that robots are incompetent if they look like a baby.[25] Thus, a robot's facial schema can have varied impacts on HRI depending on the context. Robot's facial schema might influence HRI positively if it is congruent with the purpose of the robot. For example, the baby schema facial trait of a caretaking robot might improve HRI, while it can be a negative factor for a sanitization robot.

Emotional expression. People are more emotionally engaged with a robot if it is anthropomorphic.[6] Moreover, people also expect anthropomorphic robots to be more emotionally expressive, alive, and sociable compared to relatively less anthropomorphic robots.[10] Emotional expressions of a robot might be seen as a positive trait that can improve HRI.

However, findings on the effect of robots' emotional expressions on HRI are rather mixed. Emotional expressions of a robot might make people feel uncomfortable about interacting with it.[27] The congruence between user expectations and the robot's emotional expressions can be a factor that can explain, consciously or unconsciously, how underlying beliefs are influenced.[3] Many of the heuristics and expectations people develop in human-to-human interactions can inadvertently affect how they evaluate anthropomorphic robots.[40] Interacting with humanoids is still new to most people, and subsequently, incongruence in robot behavior and people's expectations can trigger an adverse reaction. For example, a robot's emotional expressions in a high-contact service situation such as an upscale restaurant are expected and tolerated,[36] while such expressions in a low-contact service situation, such as in a grocery store, might come off as odd and eerie.[27]

Neurophysiological measures suggest the segment of the brain that guides compassion becomes more active not only around other humans but also around anthropomorphic robots.

User expectation. Some people can be averse to robots because they are inherently more adverse to change and new things.[35] People who are referred to as laggards according to innovation diffusion theory are intrinsically unfavorable of change.[35] Subsequently, these people might experience a stronger uncanny valley effect when interacting with humanoids.[24]

Additionally, previous research suggests that people might have varying *cultural* expectations of robots. Japanese people are more open to the idea of robots performing interactive tasks such as giving a massage while Europeans expect robots to perform assistive tasks such as snow lowing.[17] To prevent biased results, future research should survey the degree of robotic acceptance and robotic expectations before evaluating human responses toward anthropomorphic robots.

Uncanny valley was first theorized by robotics professor Masahiro Mori in the 1970s, where he and his team at the Tokyo Institute of Technology observed an abrupt shift in human attitudes toward robots as anthropomorphism efforts increased.[29] People behaved favorable toward anthropomorphic robots at the outset, but as more human-like features were added, the response shifted from excitement to revulsion.[29] Indeed, such robots were viewed as a threat to humanity.

The uncanny valley theory has been supported empirically, however, people might experience varied degrees of 'uncanniness' depending on the situational context and the inherent personality. The same emotional expression of a robot can make some people more willing to interact with the robot or exacerbate the sense of uncanniness about interacting.[23,40] It is interesting to note that age is a substantial moderating factor for a sense of uncanniness from interacting with humanoids.[35] Children older than nine, as the adults did, evaluated the anthropomorphic robot to be creepier than the machine-like robot, but children younger than nine did not.[5] Furthermore, a recent study discovers empirical evidence that supports the existence of an additional uncanny valley,[22] which suggests the psychological mechanism of why people feel uncanny about anthropomorphic robots is complex. Since the uncanny valley effect is not observed among children younger than nine, there must be confounding factors that cause people to feel strange about humanoids. Insights into potential mediating factors can help us better understand the uncanny valley effect.

Moral agency. Humanoids are not moral agents. They are objects and human moral values do not apply to them.[38] However, people tend to assign human characteristics to anthropomorphic objects, and subsequently, humanoids are likely to be seen as moral agents.[40] More anthropomorphism invokes stronger empathy toward objects.[18] Humanoids that almost feel like humans should elicit a comparable level of empathy people have for fellow humans. Uncanny valley research shows that people's attitudes shift again to the positive end once the robot is anthropomorphic almost comparable to humans.[36]

Because people see highly anthropomorphic robots as moral agents and feel empathy toward these machines, the use of humanoids can also be a moderating factor for people's evaluation. For example, humanoids that work in a profession that risks life and limb for the public good, such as a counterterrorism unit and a firefighter, should be viewed positively while humanoids in professions that are not considered dignified, such as sex robots, might be seen negatively. In addition, there is no agreement on whether people are ready to have humanoids that elicit such a level of empathy.[27]

Conclusion

This article discussed the effect of robot anthropomorphism on HRI and potential moderators that might alter the effect. Robot anthropomorphism can improve HRI in terms of people's cognitive and motivational responses. Conversely, robot anthropomorphism might trigger a sense of uncanniness. To address this duality in previous findings, future research might further investigate factors that might moderate the effect of robot anthropomorphism on HRI. **C**

References

1. Aggarwal, P. and McGill, A.L. Is that car smiling at me? Schema congruity as a basis for evaluating anthropomorphized products. *J. Consumer Research 34*, 4 (2007), 468–479.
2. Bhimasta, R.A. and Kuo, P. What causes the adoption failure of service robots? A case of Henn-na Hotel in Japan. In *Proceedings of the 2019 ACM Intern. Joint Conf. Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM Intern. Symp. Wearable, (London, U.K., 2019)* 1107–1112.
3. Blut, M., Wang, C., Wünderlich, N.V. and Brock, C. Understanding anthropomorphism in service provision: a meta-analysis of physical robots, chatbots, and other AI. *J. Academy of Marketing Science 49*, (2021), 632–658.
4. Borgi, M., Coliati-Dezza, I., Brelsford, V., Meints, K. and Cirulli, F. Baby schema in human and animal faces induces cuteness perception and gaze allocation in children. *Frontiers in Psychology 5*, (2014), 411.
5. Brink, K.A., Gray, K. and Wellman, H.M. Creepiness creeps in: Uncanny valley feelings are acquired in childhood. *Child Development 90*, 4 (2017), 1202–1214.
6. Broadbent, E. et al. Robots with display screens: A robot with a more humanlike face display is perceived to have more mind and a better personality. *PLOS One 8*, 8 (2013), e72589.
7. Cann, O. Machines will do more tasks than humans by 2025 but robot revolution will still create 58 million net new jobs in next five years, 2018; https://bit.ly/45CWfW8
8. Cheng, Y., Qiu, L., and Pang, J. Effects of avatar cuteness on users' perceptions of system errors in anthropomorphic interfaces. In *Proceedings of the 2020 Intern. Conf. Human-Computer Interaction, (Copenhagen, Denmark, 2020)*, 322–330.
9. Chita-Tegmark, M., Lohani, M. and Scheutz, M. Gender effects in perceptions of robots and humans with varying emotional intelligence. In *Proceedings of the 14th ACM/IEEE Intern. Conf. Human-Robot Interaction. (Daegu, South Korea, 2019)*, 230–238.
10. Choi, S., Mattila, A.S. and Bolton, L.E. To err is human(-oid): How do consumers react to robot service failure and recovery? *J. Service Research 24*, 3 (2021), 354–371.
11. Chun, B. and Knight, H. The robot makers: An ethnography of anthropomorphism at a robotics company. *ACM Trans. Human-Robot Interaction 9*, 3 (2020), 1–36.
12. Colle, A. The role of aesthetics in robotics and the rise of polymorphic robots. *Companion of the Proceedings 2020 ACM/IEEE Intern. Conf. Human-Robot Interaction, (New York, N.Y., USA, 2020)* 623–624.
13. Diehl, J., Schmitt, L.M., Villano, M. and Crowell, C. The clinical use of robots for individuals with autism spectrum disorders: A critical review. *Res Autism Spectr Disord 6*, 1 (2012), 249–262.
14. Ellingrud, K. The upside of automation: New jobs, increased productivity and changing roles for workers, 2018; https://bit.ly/45Hw7JQ
15. Epley, N., Waytz, A., Akalis, S. and Cacioppo, J.T. When we need a human: Motivational determinants of anthropomorphism. *Social Cognition 26*, (2008), 143–155.
16. Hanson, D. Why we should build humanlike robots. *IEEE Spectrum* (2011).
17. Harling, K.S., Mougenot, C., Ono, F., and Watanebe, K. Cultural differences in perception and attitude towards robots. *Intern. J. Affective Engineering 13*, 3 (2014), 149–157.
18. Harrison, M. and Hall, A. Anthropomorphism, empathy, and perceived communicative ability vary with phylogenetic relatedness to humans. *J. Social, Evolutionary, and Cultural, Psychology 4*, 1 (2010), 34–48.
19. Hoenen, M., Lübke, K.T. and Pause, B.M. Non-anthropomorphic robots as social entities on a neurophysiological level. *Computers in Human Behavior 57*, (2016), 182–186.
20. Ito, M. Softbank's Pepper robot debuts as coffee machine salesman at Bic Camera, 2014; https://bit.ly/40c1T0b.
21. Kappes, A., Harvey, A.H., Lohrenz, T., Montague, P.R. and Sharot, T. Confirmation bias in the utilization of others' opinion strength. *Nature Neuroscience I*, (2020), 130–137.
22. Kim, B., Bruce, M., Brown, L., Visser, E. and Phillips, E. A Comprehensive approach to validating the uncanny valley using the anthropomorphic RoBOT (ABOT) database. In *Proceedings of the 2020 Systems and Information Engineering Design Symp. (Charlottesville, VA, USA, 2020)*, 1–6.
23. Kim, S. and McGill, A.L. Gaming with Mr. Slot or gaming the slot machine? Power, anthropomorphism, and risk perception. *J. Consumer Research 38*, 1 (2011), 94–107.
24. Kim, S.Y., Schmitt, B.H. and Thalmann, N.M. Eliza in the uncanny valley: Anthropomorphizing consumer robots increases their perceived warmth but decreases liking. *Marketing Letters 30*, (2019), 1–12.
25. Maeng, A. and Aggarwal, P. Facing dominance: Anthropomorphism and the effect of product face ratio on consumer preference. *J. Consumer Research 44*, 5 (2018), 1104–1122.
26. Markets and Markets Analysts. *Service Robots Market* (2021); https://bit.ly/3MfpwPL.
27. Mende, M., Scott, L., van Doorn, J., Grewal, D. and Shanks, I. Service robots rising: How humanoid robots influence service experiences and consumer responses. *J. Marketing Research 56*, 4 (2019), 535–556.
28. Mitchell, W., Szerszen, K., Lu, A., Schermerhorn, P., Scheutz, M. and MacDorman, K. A mismatch in the human realism of face and voice produces an uncanny valley. *I-Perception 2*, (2011), 10–12.
29. Mori, M. The uncanny valley. *IEEE Spectrum* (2012).
30. Mourey, J.A., Olson, J.G., and Yoon, C. Products as pals: Engaging with anthropomorphic products mitigates the effects of social exclusion. *J. Consumer Research 44*, 2 (2017), 414–431.
31. Natarajan, M. and Gombolay, M. Effects of anthropomorphism and accountability on trust in human-robot interaction. In *Proceedings of the 2020 ACM/IEEE Intern. Con. Human-Robot Interaction, (Cambridge, U.K., 2020)*, 33–42.
32. Phillips, E., Zhao, X., Ullman, D. and Malle, B.F. What is human-like? Decomposing robots' human-like appearance using the anthropomorphic roBOT (ABOT) database. In *Proceedings of the 2018 ACM/IEEE Intern. Conf. Human-Robot Interaction*. ACM, New York, N.Y., USA, 105–113.
33. Pixteren, M. et al. Trust in humanoid robots: Implications for services marketing. *J. Service Marketing 33*, 4 (2019), 507–518.
34. PwC. Will robots really steal our jobs? An international analysis of the potential long-term impact of automation, 2018.
35. Rangaswami, A. and Gupta, S. Innovation Adoption and Diffusion in the Digital Environments: Some Research Opportunities. *New Product Diffusion Models*. Springer, New York, NY, USA, 2020.
36. Roesler, E., Naendrup-Poell, L., Manzey, D. and Onnasch, L. Why context matters: The influence of application domain on preferred degree of anthropomorphism and gender attribution in human–robot interaction. *Intern. J. Social Robotics 14*, (2022), 1155–1166.
37. Sheehan, B., Jin, H.S. and Gottlieb, U. Customer service chatbots: Anthropomorphism and adoption. *J. Business Research 115*, (2016), 14–24.
38. Torrance, S. Ethics and consciousness in artificial agents. *AI & Soc 22*, (2008), 495–521.
39. Ullman, D. and Malle, B.F. What does it mean to trust a robot? Steps toward a multidimensional measure of trust. *Companion of the Proceedings 2018 ACM/IEEE Intern. Conf. Human-Robot Interaction, (New York, N.Y., USA, 2018)*, 263–264.
40. Visser, E.J. et al. Almost human: Anthropomorphism increases trust resilience in cognitive agents. *J. Experimental Psychology: Applied 22*, 3 (2016), 331–349.

Rae Yule Kim is an assistant professor at Monclair State University, Monclair, NJ, USA.

DOI:10.1145/3624719

A Google case study finds ML training in the cloud can reduce CO_2e emissions up to 100×.

BY DAVID PATTERSON, JEFFREY M. GILBERT, MARCO GRUTESER, EFREN ROBLES, KRISHNA SEKAR, YONG WEI, AND TENGHUI ZHU

Energy and Emissions of Machine Learning on Smartphones vs. the Cloud

GLOBAL CLIMATE CHANGE is a huge challenge facing society today. The rapid growth of computing overall and of machine learning (ML) in particular rightfully raises concerns about their carbon footprints. As an early and enthusiastic adopter of ML, a manufacturer of millions of smartphones annually, and a significant cloud provider, Google is in a nearly unique position to compare the impact and efficiency of ML on the two ends of the information technology (IT) computing spectrum.

This article makes four main observations:

1. Most energy studies of smartphones ignore chargers, but they surprisingly use ~3× as much energy as the smartphones themselves, with the main culprits being the use of multiple chargers per phone and the inefficiency and increasing popularity of wireless chargers. Compared to cloud datacenters, this energy overhead (*charger power usage effectiveness*) is ~2.9× worse.

2. Smartphones can charge at off-peak hours, and many ML workloads can also shift to run at off-peak times. However, the cloud also lets ML experts shift to remote sites with much greener energy, while phones predominantly must use whatever the local utility provides. Given the 2.9× higher energy overhead, ML training on devices can have up to ~25× the carbon emissions as doing it in the cloud even if the computation energy was the same.

3. One example suggests ML training on smartphones uses up to 12× the computation energy of datacenter training. There are privacy and other advantages for training on devices,

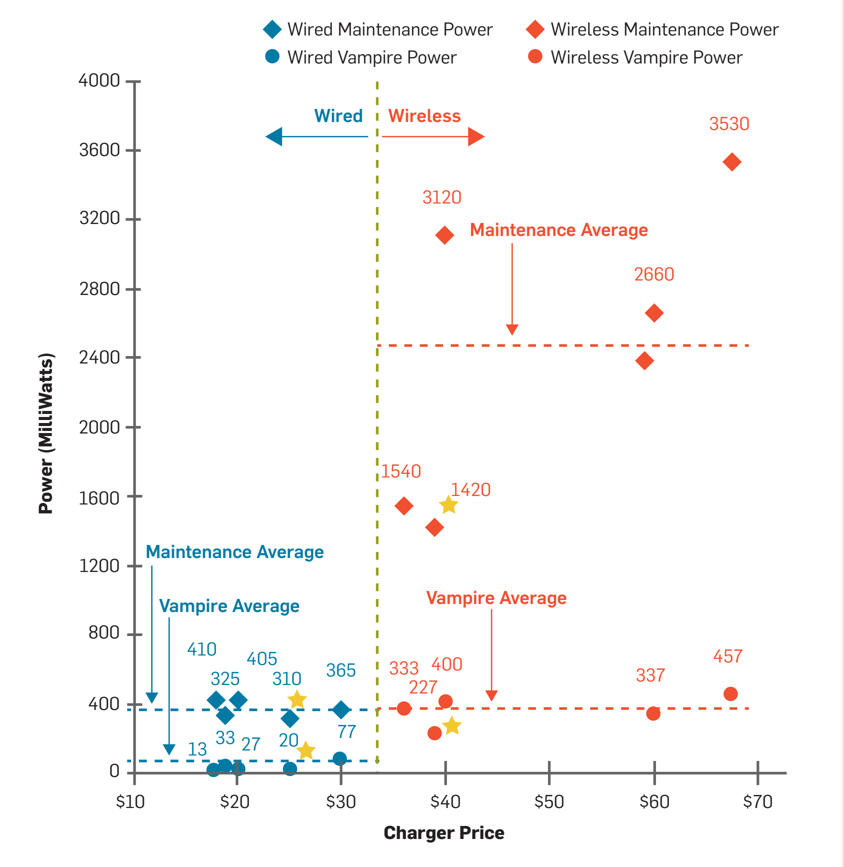
» key insights

- **Smartphone charger inefficiency is a much larger energy consumption issue than ML on smartphones. Chargers were responsible for 70% of energy use while ML was <3%.**
- **While training ML models on smartphones has inherent advantages for privacy, it can have 100× the carbon footprint of training in the cloud.**
- **In 2021, ML energy use was <15% of Google datacenter and <3% of smartphones. The bigger carbon challenge for IT is likely the embodied carbon footprint from manufacturing computers in general and smartphones in particular. The embodied carbon footprint of the short-lived smartphones in 2021 was nearly 3× that of datacenter servers. Despite their large, embodied carbon cost, we have discarded 7.5 billion smartphones in the recent past.**

CORRECTED VERSION OF RECORD. V 1.1. PUBLISHED JANUARY 25, 2024


LIGHTING
OUTDOOR USE
No. AE 29588573

Figure 1. Average measured maintenance mode power (diamonds) and vampire power (circles) for the average of a Pixel 5 and a Pixel 6 Pro versus price for five wired chargers (blue on the left) and five wireless chargers (red on the right). Gold stars tag the most efficient chargers over a full day.



but combining 12× with the previous observation means *carbon dioxide equivalent emissions* (*CO_2e,* including greenhouse gasses) can be two orders of magnitude higher.

4. From 2019 to 2021, ML represented between 10% and 15% of the total annual operational energy use in the Google cloud (with ⅔ of that for training) and less than 3% of the total annual operational energy use on smartphones (with 1/100 of that for training) in 2021. The major climate change challenge for IT is elsewhere, likely the *embodied* CO_2e from manufacturing computers.

Keep in mind this article is *not* a comparison of *all* computation done on phones and the cloud, but solely on the impact of ML on energy use and operational CO_2e. We provide the data to support these insights. While primarily focused on *operational* CO_2e generated from computer use, we also address the relative impact of embodied CO_2e.

How Significant Is Charger Energy Use Compared to Smartphones Themselves?

Computers in datacenters draw electricity from the grid continuously. Because smartphones operate from a battery, they only draw electricity from the grid when connected to a charger. To account for smartphone ML energy accurately, we must include the energy overhead of their chargers. Smartphones are charged two ways:

- *Wired charging* via a cable and an AC/DC adapter, and
- *Wireless charging* via inductive coils in addition to the AC/DC adapter.

Wireless charging is increasingly popular due to its convenience and the reduction in smartphone wear and tear by avoiding the repeated insertion of a cable.

For wired charging, energy is lost from the AC/DC power adapter in the charger and in the power management integrated circuit (PMIC) battery charger in the phone. Wireless charging loses additional energy through the inductive coils. The percentage of energy going to the smartphone is then:

$$\textit{Wired charging efficiency} = \textit{adapter efficiency} \times \textit{charging efficiency}$$

$$\textit{Wireless charging efficiency} = \textit{adapter efficiency} \times \textit{coils efficiency} \times \textit{charging efficiency}$$

For conventional designs, the efficiencies are 80% for the adapter, 80% for the inductive coils, and 90% for the battery charging circuitry. The 2018 USB Power Delivery 3.0 standard offers a *programmable power supply* (PPS) that improves efficiency: 90% for the adapter, 83% for the inductor coils, and 97% for the battery charging circuitry.

Using the formulas here, the overall charger efficiencies are 58% (80% × 80% × 90%) for non-PPS wireless chargers, 72% (90% × 83% × 97%) for PPS wireless chargers, 72% (80% × 90%) for non-PPS wired chargers, and 87% (90% × 97%) for PPS wired chargers. Thus, wireless chargers are ~1.25× less efficient while charging phones.

The charger uses sensors to detect when even an uncharged phone is connected to it. In addition to the smartphone charger efficiency when it is in use, we also need to account for *vampire power usage*. Also called *standby power*, it is the power wasted by the charger when not connected to a smartphone. Vampire power adds up and can be a significant amount of energy consumed.

A final factor is *maintenance mode power*, the power consumed when the phone is fully charged but it is still connected to the charger. Depending on the charger, it can still draw significant energy even when the smartphone does not need any more energy.

The total energy consumption of a smartphone and charger is then:

$$\textit{Average smartphone and charger energy usage} = \frac{\textit{Average smartphone energy}}{\textit{Overall efficiency}} + \textit{vampire energy} + \textit{maintenance energy}$$

A further complexity is that many smartphone users have multiple plugged-in chargers. One survey found that 23% of users had only one charger, but 39% had four or more.[38] If we assume the 39% had exactly four chargers, the average of that survey is 2.7 chargers per smartphone.

Datacenters use the industry standard metric of *power usage effectiveness* (*PUE*), which accounts for the energy "wasted" that goes into the datacenter but is used for power distribution and cooling instead of powering the computers in the datacenter. If the energy overhead was 50%, the PUE would be 1.50. The global average datacenter PUE[28] is ~1.60, while cloud datacenter PUEs are typically ~1.10.

We propose a new metric to quantify the efficiency of a smartphone with its chargers analogously to the datacenter PUE, where the goal is to try to get as close to 1.00 as possible. We define *charger PUE* (*CPUE*) as

$$\frac{\textit{Average Smartphone and Charger Energy} + \textit{Vampire Energy Chargers}}{\textit{Average Smartphone Energy}}$$

where *vampire energy chargers* is the energy consumed by chargers plugged in but unused.

Google measurements supply some of the parameters needed to calculate CPUE:

- The average wireless charger use was 42% and 58% for wired chargers, with the wireless percentage higher for newer phones and lower for older phones.
- Average time to fully charge was 107 minutes (1.8 hours), like EPA's 2-hour estimate.[12]
- Average hours in maintenance mode were 302 minutes (5.0 hours), close to a 2011 study that reported 4.6 hours.[30]

We also purchased and measured five wired and five wireless chargers both from smartphone manufacturers (Apple, Google, Samsung) and from third-party suppliers (Anker, Belkin). To test maintenance mode, we used the average of a fully charged Pixel Pro 6 and a fully charged Pixel 5. They were turned off, so that no applications could run during maintenance mode. If apps were left on—which might be the more typical case—maintenance mode power would be even higher, so these numbers might be conservative. All chargers were measured using a Poniie PN1500 portable power meter, which is accurate to ±10 milliwatts and has a range down to 20 milliwatts.[a]

Figure 1 plots the individual measurements versus their price. Our findings:

- Wireless chargers are more expensive and use more power than wired chargers; their overall average energy use over 24 hours is 2.4× higher.
- Surprisingly, money doesn't buy efficiency. The most power efficient wired charger is Google's and Apple sold the wireless charger winner. They use 40%–60% less energy but are 20%–70% cheaper than the high-priced ones.
- Average vampire mode power for wired chargers is 34±25 milliwatts and 363±45 for wireless chargers (~11× more than wired).
- Average maintenance mode power for wired chargers is 361±86 and

a To ensure accuracy, we bought three copies of each charger, but they varied only by power-meter accuracy. Maintenance power was measured after several hours to ensure the device reached steady state and the battery was fully charged, consistent with the U.S. DOE Uniform Test Method.[11] We double-checked the wireless chargers' measurements using a Ketotek (KTEM02) power meter, which averaged within 3%.

Table 1. Daily smartphone energy use and charger PUE for each charger option and charger PUE assuming 2.7 chargers with 42% being wireless and 50% using PPS.

	Wired		Wireless	
	No PPS	PPS	No PPS	PPS
Smartphone (Wh)	9.28			
Charger charging (Wh)	3.61	1.35	6.83	3.53
Maintenance mode (Wh)	1.82		12.35	
Vampire if used (Wh)	0.58		6.24	
Vampire if unused (Wh)	0.82		8.71	
Avg. charger energy (Wh: 1 charger)	6.01	3.75	25.42	22.12
Charger PUE (1 charger)	1.65	1.40	3.74	3.38
Average charger energy (Wh: 2.7 chargers, 42% wireless, 50% PPS)	19.84			
Charger PUE (2.7 chargers)	3.14			

Table 2. Charger PUE sensitivity as parameter values vary (halving/doubling or ± standard deviation). The average and standard deviation were calculated across 20 measurements of Pixel 5 and Pixel 6 Pro phones.

	Value			CPUE	
Item	Nominal	Low	High	Low	High
Number of Chargers	2.7	1.35	5.4	2.54	4.34
Wireless %	42%	21%	84%	2.41	4.60
Wireless Maintenance Mode (mW)	2454	1512	3396	2.92	3.35
Wireless Vampire Mode (mW)	363	318	408	3.02	3.26
Power Supply (nonPPS)	50%	25%	100%	3.07	3.28
Wired Maintenance Mode (mW)	361	275	447	3.11	3.16
Wired Vampire Mode (mW)	34	9	59	3.05	3.23

2454±942 for wireless (~7×). Thus, the average *wireless vampire* power matches the average *wired maintenance* power.

Using these values for the parameters in the equations here, Table 1 calculates the smartphone CPUE by charger and phone type and then summarizes aggregate information for a mix of 2.7 chargers per phone, 42% wireless, and 50% PPS. The higher vampire power usage of wireless chargers is tenfold that of wired chargers. Given these assumptions on the portion that are PPS, the portions that are wireless, and number of chargers, CPUE is ~3.1. That is, charger overhead energy is ~2.1 times what a smartphone uses.

To test the sensitivity to these values, we calculated CPUE by varying each assumption individually either by subtracting or adding the standard deviation to the average or halving or by doubling a parameter while keeping the other parameters constant. Table 2 shows the results. CPUE is most sensitive to the percent of wireless chargers (2.41–4.60) followed by the number of chargers per smartphone (2.54–4.34) and wireless maintenance mode power level (2.92–3.35). For this wider range of parameters, CPUE is at least ~2× to as much as ~5× the energy consumption of smartphones by themselves.

How Do Cloud Energy Use and Carbon Emissions Compare to Those for On Device?

The 4Ms determine the carbon footprint of training in the datacenter:[28] Model, machine, mechanization (datacenter PUE), and map (input energy carbon intensity per geographic location). Presuming we train the same model on the datacenter and on device, the efficiency of the three other Ms determines their relative carbon footprint. The CPUE of 3.1 for smartphone chargers is 2.9× the typical 1.10 PUE of cloud datacenters. The global average energy carbon intensity is appropriate for the 6.6B smartphones. However, the cloud allows training to be moved to datacenters with a high percentage of *carbon free energy (CFE),* which means using renewable energy (for example, solar or wind) or nuclear

Even if the energy consumption for the computation was the same in the cloud and on device, the carbon emissions could be ~25 times higher on device.

energy. In 2021, the average CFE portion was ~40% for U.S. and Europe and ~30% for Asia.

The global average conversion factor from electricity to CO_2e is 0.475kg/kWh. Patterson et al. found that training in Oklahoma reduced carbon emissions by >6× (to 0.074) versus the global average due to its 93% CFE, as Google acquired renewable energy in Oklahoma matched on an hourly basis with its energy consumption.[28] Thus, even if the energy consumption for the computation was the same in the cloud and on device, the carbon emissions could be ~25 times higher on device.[b]

How Does Energy Consumed for ML Training in Cloud Datacenters Compare to Training on Smartphones?

Comparing *federated learning* to datacenter training is challenging because the inherently more limited resources on a phone typically lead to different model designs and training approaches, which together with differences in the training data distribution and accessibility lead to quality differences.

Despite these challenges, we thought it would be interesting to get a ballpark estimate of differences between training on the datacenter and on device. As a case study, we compare the energy consumed for training an identical model per 10M training examples processed in the datacenter and in a federated setting. We identified a prototype ~10,000 parameter multi-layer perceptron model used for semantic location inference, which is the process of inferring a device's location in terms of a place as defined in the Android Places SDK (for example, local business or point of interest) instead of in coordinate space. We ran it on TPUv4 lite, and the average energy of three runs was 59 kilojoules.

The TPUv4 lite chip[23] trains this workload in under two minutes. This experiment has its limitations:

b In case CFE is unclear, Google offsets its CO_2e by buying an equal amount of carbon reduction at other locations later, usually in renewable energy companies. We *ignore* those offsets in this article for CFE. If we instead made that assumption, then ML in the cloud would have zero operational CO_2e while smartphones would remain as is, since no organization offset their CO_2e in 2021.

▸ The smartphone workload was not tuned to the TPU. It is currently I/O bound, so the TPUv4 lite chip is nearly idle at under 1% utilization.

▸ Moreover, despite trying to compute on only one host CPU, the dual CPU host server of TPUv4 lite uses 86% of the 47 kilojoules for the workload. This server is overkill for this application.

▸ However, this energy calculation omits the energy to process the raw data in the datacenter. We also assumed 10M examples are sufficient to train for batch size 64, but we didn't check for convergence.

We suspect that tuning the workload to reduce I/O bottlenecks and properly configure the host hardware would reduce datacenter energy consumption significantly.

For federated learning, the average user holds 16 training examples with typically 10 local epochs, so 160 examples are processed per device session. This ratio yields 62,500 device training sessions per 10 million training examples processed. Assuming typical energy consumption figures, Table 3 shows an energy use of 186 kilojoules on the smartphone and 381 kilojoules in the communication and serving infrastructure for a total of 567 kilojoules.

This estimate for on device energy is also limited:

▸ The model uses a relatively low number of training examples per user device, which results in an even larger number of device training sessions. With more training data per device, per session costs would be better amortized and we would expect federated learning overhead to decrease significantly.

▸ The five seconds for processing reported through our telemetry includes additional waiting time and other lower energy states.

▸ The federated learning server figure of 6J/device session represents an average over all models and is likely an overestimate for this small model.

▸ It doesn't account for the difference in data transfer to the datacenter of all the raw data versus federated learning only needing to access the portion of the data needed for the calculation.

▸ However, it omits the relatively large fixed-energy cost of the broadband infrastructure (and wireless access point), since it typically remains on regardless of whether federated learning or other communications are active. Including amortized fixed costs increases federated learning energy use ~4× to 2.3 megajoules.

Although it is difficult to make a fair and detailed comparison between a training task done on smartphones and in the datacenter, the estimated ~567 kilojoules for federated learning represents ~12× of the estimated ~47 kilojoules for datacenter training. For the reasons given here, this observation is only a ballpark estimate.

In addition to the ~12× energy increase over the datacenter, the ~25× overhead in emissions due to the higher PUE and higher carbon content energy of on-device computation suggests the datacenter is the much lower emissions option for applications where the data processing location is flexible.

Table 3. Federated learning energy usage for the case study semantic location model.

	Power/Energy Consumption	Measurements for Model	Energy (J)
Phone Wakelock	<100mW	5s	0.5
Phone Communication (Wi-Fi)	130nJ/bit	250KB	0.27
Phone Computation (CPU)	440mW	5s	2.2
Subtotal (Phone)/Device Session			2.97
Fixed Broadband (variable cost)	0.5W @ 10Mbps = 50nJ/bit	250KB	0.1
Federated Learning server	6J/device session		6
Subtotal (Infra)/Device Session			6.1
Device Sessions/10M examples	62,500		
Phone/10M examples			185,625
Infra/10M examples			381,250
Total			566,875

Table 4. Energy consumption of four smartphones. The first three columns were published earlier.[1,9,22]

Smartphone	Neo Freerunner	iMate Kjam	HTC HD2	Pixel 6	Average
SoC	33%	35%	37%	40%	36%
Wireless	44%	57%	26%	23%	37%
Display	9%	3%	26%	29%	17%
Rest	14%	5%	10%	9%	10%

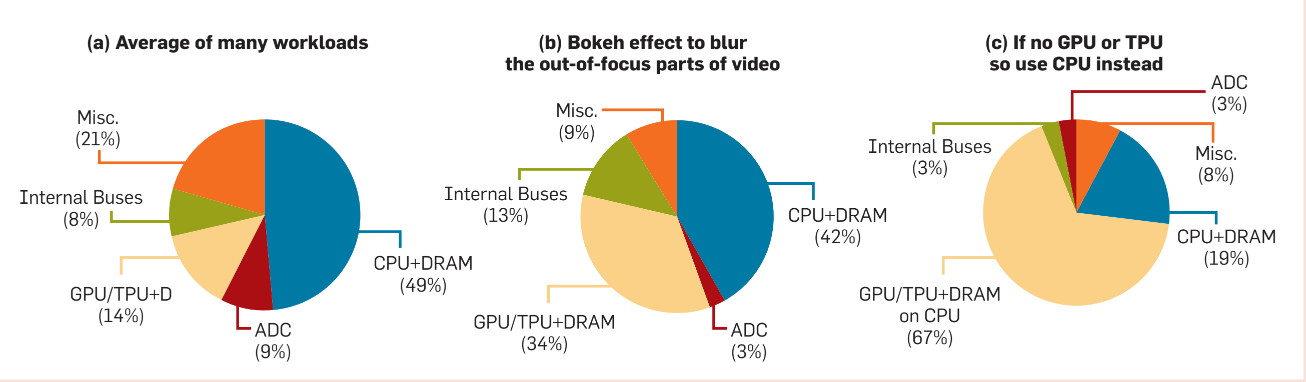
How Does ML Energy Use in Cloud Datacenters Compare to Its Use on Devices?

Patterson et al. showed that between 10% and 15% of Google's overall energy usage in datacenters was for ML for 2019–2021.[28] The next step to answer this section's question is to see where the energy goes in smartphones.

Table 4 shows the energy breakdown of four smartphones between their major components: the System on a Chip (plus DRAM energy), wireless, display, and the rest (including audio and camera). A rule of thumb from Table 4 is that little more than ½ of the energy is for wireless and the display, a little more than ⅓ is for the SOC, and the rest use about $^{1}/_{10}$.

The main SoC is the biggest integrated circuit in a smartphone and contains CPUs, memory interfaces, and domain specific processors such as GPUs and ML accelerators. The Edge Tensor Processing Unit (TPU) is Google's ML accelerator, similar in

Figure 2. Energy consumption of the main Pixel 6 SoC for (a) an average of many workloads; (b) a workload that blurs out-of-focus parts of video images (Bokeh effect); and (c) an average use if there was no GPU or TPU so their computation must be done on the CPU (assuming a 12.5x difference in performance/Joule). This GPU/TPU replacement also increases SOC energy consumption by 2.6x. DRAM energy is distributed between the CPU, GPU, and TPU. AOC is the Always On Compute controller.



spirit to the Apple Neural Engine and the Samsung NPU.[4,21]

Understanding the energy for ML requires examining the SoC energy. Figure 2a breaks down its energy consumption to the primary power using components for the Pixel 6 SoC for the average of many workloads. This data was collected running workloads in our labs, which are consistent with measurements of smartphones in the field.

One might wonder if the GPU and TPU are important since the CPU uses more than three times as much energy. The average use belies their value for three reasons.

First, smartphone applications can be limited either by long-term average energy use (which drains the battery), instantaneous energy use (to prevent overheating), or by latency (impacting user experience); the GPU and TPU help with all three. Even if on average they don't use as much energy, they enable applications that would otherwise be infeasible on a smartphone CPU. For example, Google Camera ambient recognition of visual entities (QR codes, text, objects, among others) allows rapid detection and proactively highlights so the user may act on them. The Edge TPU keeps latency low, so the system remains interactive and energy consumption low to avoid thermal problems. The performance/Joule improvement of the Edge TPU over the CPU for this application is 12.5×.

Second, given billions of phones and users, smartphone engineers design for many workloads, not just one. Figure 2b shows the GPU and TPU are ⅓ of one video workload.

Finally, Figure 2a shows energy use *after* the application was accelerated; it doesn't show what the energy use would be if it was all done on the CPU. Assuming the TPU and GPU provided 12.5× performance/Joule over the CPU, SOC energy would increase 2.6× and their portion would now be ⅔ of the energy, as Figure 2c shows.

To answer our question about the energy use, 36% is for SOCs (Table 4). Although most ML uses accelerators and the GPUs, some still use CPUs. Most CPU workloads are for the popular non-ML use cases such as social media, video streaming, browsing, messaging, and so on. An exact CPU percentage for ML is difficult to determine, but an upper bound is easier to estimate. We assume ~10% of CPU energy and ~25% of the GPU and TPU energy is for ML inference, which are generous upper bounds. Our upper bound estimate for ML is:

$$= \sim 36\% \times (\sim 10\% \times 49\% + \sim 25\% \times 14\%)$$

$$= \sim 3.0\%$$

The answer to the question at the outset of this section is that ML represents <3% of smartphone energy use and 10%–15% of datacenter energy use.

Table 5. Daily federated learning energy usage on smartphones for representative models.

Model	Gboard	Model B	Chat
Phone Wakelock	9J	1J	15J
Phone Communication	26.3J	0.6J	2.1J
Phone Computation	39.6J	2.2J	22J
Phone Energy/device session	75.5J	3.8J	39.1J
Infra Energy/device session	16.4J	6.2J	6.8J
Device sessions/day	150K	600K	14K
Energy per day (across all devices)	~=11.3MJ (3.1kWh)	~=2.2MJ (0.6 kWh)	~=0.5MJ (0.1kWh)
Energy per day (infra + all devices)	~= 13.8MJ (3.8kWh)	~= 6MJ (1.6kWh)	~= 0.6MJ (0.2kWh)

What Portion of Energy Is for ML Training versus Inference?

Patterson et al.[28] found the ML energy use in Google datacenters was 40% for training and 60% for inference. We now calculate the split for smartphones.

Google uses a new federated learning system that trains ML models on devices. One advantage of federated learning is that it doesn't require shipping data to the datacenter, which preserves privacy.

To account for real-world conditions, we estimate the energy consumption of an average device session using telemetry from Google's federated learning system while this model is training on a fleet of Android devices. A federated learning training session involves four steps:

1. Connecting with and waiting for the server to signal the start of a training round;
2. Downloading model parameters;
3. Setting up and executing training computations; and,
4. Uploading the model update.

We therefore expect the dominant factors on the smartphone are communication, computation, and to some extent the wakelock drain while waiting for the server to start a training round. Hence, we model training session energy consumption as follows:

$$\begin{aligned}&\textit{Training session energy consumption}\\&= \textit{wakelock drain} \times \textit{waiting time}\\&+ \textit{WiFi energy/bit} \times\\&\quad(\textit{bits downloaded} + \textit{uploaded})\\&+ \textit{CPU Consumption} \times \textit{training time}\end{aligned}$$

Communication energy consumption varies widely based on the amount of ML model parameters that must be transferred and the quality of WiFi link. We use an energy/bit model to account for the varying data transfer sizes and choose an average energy/bit figure of 130nJ/bit based on reported measurements.[35] Note that from high to low link quality, WiFi energy/bit can change by about two orders of magnitude.

To estimate the total energy consumed in federated learning, we studied the daily energy consumption of three models that we believe to be representative of models currently being trained in Google's production federated learning system. Table 5 summarizes the results. To put individual smartphone energy usage for ML training into perspective, the range observed here generally requires a few seconds of additional phone charging time.

Since 2021, Google has been reducing CO_2e using the flexibility of the cloud to shift computation in both location and time, which is even more effective at reducing emissions.

Using the highest smartphone energy estimate of the three sample models (75.5J per device session) and the count of device sessions per day from our telemetry, this yields an upper bound of 11.3GJ or 3.1MWh per day, which represents less than 0.01% of the worldwide smartphone-only operational energy usage (not including chargers). We believe that even this estimate is a generous upper bound for federated learning since a significant fraction of these device sessions represent analytics (non-ML) workloads.

The answer to the question of this section is that training represents ~40% of ML energy use in datacenters but on smartphones it is only ~0.3%, a factor of ~133x less.

Discussion

We now discuss a few insights from our investigation.

Does shifting the time of charging make smartphone-charging energy use clean? Some might think that shifting the time of charging made smartphone charging "clean." This perspective may have been influenced by a new feature of the iOS 16.1 called "clean energy charging."[4] The iPhone uses forecasts of the CO_2e in the local grid to charge an iPhone during times of cleaner energy production. Although limited to the US, it likely reduces CO_2 a bit there. However, even within the U.S. there are many grids where the energy is 10X dirtier than others, no matter the time of day. Since 2021 Google has been reducing CO_2e using the flexibility of the cloud to shift computation in both location *and* time,[24] which is even more effective. Shifted examples are cloud training and offline inference. Finally, Table 2 points out the major use of energy when there is no phone on the charger—due to redundant chargers and due to vampire mode—which is unaffected by when a phone is charged.

What is the cost of making the cloud as privacy preserving for ML as smartphones? Some might wonder if we should burden the cost of the cloud to be as privacy preserving as keeping data local to smartphones? Security experts[14,30] commented that:

▸ Today it would be many orders of

magnitude more expensive;[c]

- Sharing data is a matter of trust no matter the promises made;
- Distributed training still sends information learned to a centralized server in the cloud; and
- Sending the ML model to thousands of smartphones would expose the model and its weights (trained on private phone data) if any phones were compromised.

Trusted execution environments such as enclaves, which are already being deployed, may offer a path forward.

Reducing the energy of wireless chargers. A primary consideration in designing the wireless charging systems is the feedback loop notifying the user that the device is charging. To search for a smartphone, a digital ping is sent periodically. There is an inverse relationship between the ping frequency and the idle power consumption of the charging system. Therefore, system trade-offs are made to stay within a power budget (250–500mW) and provide the least amount of latency (< 1 second). Another major design consideration is the overall bill of material cost for the wireless charger. In the consumer electronics industry, the competition is high and there is a concerted effort to drive down the overall cost, pressuring the design team to remove as many extra components as possible. This competition makes it more challenging to include features as a bonus that optimize energy efficiency.

Nevertheless, we were surprised how much higher the CPUE was for wireless than wired chargers. The wireless charger consumes large vampire power on average 17 hours a day even when the charger is empty.

A simplistic alternative would simply be to add an on/off button or a weight sensor that detects a smartphone to a wireless charger that controls power to the inductive coils. One estimate is one billion wireless powered smartphones in 2021.[7] If hitting the button or sensing that there is no phone dropped vampire power usage to match wired chargers on all wireless chargers, turning off the coils could save ~5.6 terawatt-hours and ~2.7M t CO_2e annually. Our upper-bound estimate for ML on smartphones at 3.1 terawatt-hours is ~½ of these potential wireless charger energy savings.

The U.S. Environmental Protection Agency and the European Union set requirements for wired chargers that limit maximum vampire power. Perhaps they should investigate standards that reduce vampire power for wireless chargers as well?

The U.S. Environmental Protection Agency and the European Union set requirements for wired chargers that limit maximum vampire power. Perhaps they should investigate standards that reduce vampire power for wireless chargers as well?

Energy use and CO_2e emissions from smartphones. The 24-hour energy consumed by a typical smartphone is 9.28 watt-hours.[10] One recent estimate is 6,648M smartphones.[5] The total annual energy use by smartphones is then:

$$6{,}648{,}000{,}000 \times 365 \times 9.28wh = 22.5TWh.$$

Google's electricity usage in 2021 was 18.3TWh or ~80% of worldwide smartphone energy use.[3]

Using the conversion factor of 4.75×10^{-4} metric tons of CO_2 equivalent emissions (CO_2e) per kilowatt-hour,[20] the total annual emissions of smartphones is then ~10.7 megatons of CO_2e. Adding our CPUE estimate of 3.1, the total is ~33.6 mt CO_2e.

To put that total into perspective, it is the equivalent to ~7.5M gasoline-powered passenger vehicles driven for one year. There are ~1500M cars in the world today, so cars produce ~200× the CO_2e of smartphones.

The ML portion is <3%, so it is <1.0mt of CO_2e. Google's overall operational CO_2e (location-based Scope 2) in 2021 was 6.6mt,[15] and <15% is <0.97mt CO_2e for ML

Comparing operational to embedded CO_2e. If the operational CO_2e from ML is a modest contributor to global climate change, then what is the biggest climate change issue for IT? Our suspicion was that it is the embodied cost of IT.

The following computers were shipped in 2021:

- Smartphones: ~1,350,000,000,

c Not sharing your data with the cloud is equivalent to having a trusted computing base of size "Zero" in the cloud, that is, one must count on exactly Zero cloud components for security to hold. The only way to achieve this level of robustness while using the cloud is fully homomorphic encryption (for privacy) plus additional cryptographic mechanisms (for example, succinct non-interactive argument of knowledge) to detect tampering. This approach would be extraordinarily expensive.

- PCs: ~340,000,000,
- Servers: ~12,000,000.

Note that the numerous smartphones have the shortest lifetimes at only 2–3 years.

While embodied CO_2e costs are not yet routinely published for all computers, we use the following approximations based on studies of smartphones and PCs and the limited data available for servers:

- Smartphones: ~50kg CO_2e,[25]
- PCs: ~200kg,[25]
- Servers: From ~1000kg to ~4000[19,26,33]

To put embodied CO_2e into perspective, a typical smartphone weighs 0.2kg, so the CO_2e from manufacturing is ~250× larger. The ratio for PCs is ~75× and it is ~150× for servers.

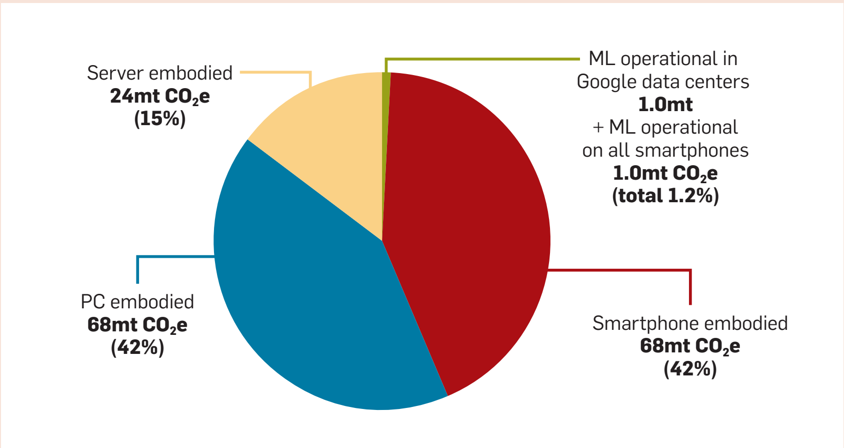
Figure 3 shows our ballpark estimate for the operational CO_2e from ML in Google datacenters and all smartphones in 2021 compared to our estimate of the embodied CO_2e for smartphones, PCs, and servers manufactured in 2021. Embodied CO_2e for information technology may be two orders of magnitude larger than the operational CO_2e from ML.

Note that Figure 3 is a snapshot of what happened in a single year. The conventional approach to comparing embodied and operational CO_2e is to predict the operational CO_2e for the equipment of a given year over the lifetimes of each computer, from 2–3 years for smartphones and 6–8 years for servers. We use the single year snapshot since we are trying to capture only the ML operational CO_2e, which we can more easily estimate what it was for 2021 rather than predicting what the ML portion will be for the next 3–8 years.

While we account for all smartphones, we don't for all datacenters. Google is one of the top three cloud providers, and it was an early adopter of ML, so its ML use might be larger than most. If we were to multiply the datacenter CO_2 portion in Figure 3 by 10× to try to account for all datacenters, ML would still only be ~7%.

Taiwan, South Korea, and Japan, with their low CFE, manufactured most of the chips in Figure 3. Changing the location of manufacture could potentially reduce CO_2e, as half is from electricity use.[36] Berger et al. predict

Figure 3. Embodied CO_2e from IT equipment versus operational CO_2e for ML in 2021. (It does not evaluate CO_2e from all computations, only ML). For smartphones, embodied CO_2e is ~70x larger than operational CO_2e for ML in 2021. Embodied server CO_2e was ~115x larger than ML operational CO_2e in Google datacenters in 2021.



decarbonizing the manufacturing supply chain of computers will take decades and cost hundreds of billions of dollars.[6] Nevertheless, the imbalance between embodied CO_2e and ML operational CO_2e in Figure 3 is so large that even if we could go back in time and use much higher CFE to reduce embodied CO_2e in 2021 by 5×, it would still be ~15× higher than operational CO_2e from ML.

Is operational CO_2e from ML one of the largest climate change challenges in information technology? The success and popularity of ML rightfully raised concerns about its environmental impact. Some estimates of CO_2e from ML were so alarming—for example, training a model in 2024 could cost $100B and produce as much CO_2e as New York City does in one month[37]—that we wanted to investigate to see what could be done to help. That inquiry led to this article and the work explored in Patterson et al.[28]

To our surprise, we found some ML papers overestimated CO_2e by >100,000×.[28] Despite considerable attention paid to the environmental impact of the operational cost of ML training, Figure 3 illustrates that embodied costs of manufacturing IT equipment might have been a ~70× larger climate change problem in 2021 than the real operational CO_2e from ML. If some prior claims about ML training were accurate, then it would have been reversed.

It's critical to use accurate estimates of CO_2e to ensure we focus on big problems. The flawed estimates on operational CO_2e from ML training that were off by >100,000× led many interested in climate change to focus on ML rather than on more prominent challenges. This article's data highlights that the embodied costs of manufacturing IT equipment may have been >70× larger than operational ML emissions in 2021, so even modest gains on embodied hardware CO_2e could easily eclipse large gains on operational ML CO_2e. The short-lived smartphones are particularly glaring, as despite having an embodied carbon footprint nearly 3× that of servers in 2021, we have discarded 7.5 billion of them in the past five or so years.

Related Work

A flurry of papers evaluated the energy use of smartphones a few years after the iPhone was announced. Carroll and Heiser[9] ran microbenchmarks to determine the energy use of hardware components as the SoC clock rates varied. They found that most of the power consumption was for the wireless radio module (GAM) and the display while the RAM, audio, and flash memory subsystems used the least. Ferreira et al.[13] did a detailed energy measurement per smartphone function and found that the top six biggest energy users were downloading data using 3G, downloading data using WiFi, sending an SMS text, making a voice call, playing an MP3 file, and the display backlight. Later papers did similar calculations for other phones.[1] Perrucci et al.[29] analyzed users' battery charging behavior to assess how often users demonstrate less than optimal charging behavior.

Since 2011, an abundance of papers have been published on smartphone energy issues, many simply surveying all the related literature. For example, Hoque et al.[18] reviews energy profilers for mobile devices stating their limitations and challenges. Javed et al.[22] survey different factors that consume energy in a smartphone. Zaman and Almusalli summarize 17 hardware and software enhancements that reduce energy consumption and the size of their benefits according to published research. These three papers survey 84, 96, and 41 publications, respectively. The most recent survey[31] tops its predecessors by citing 418 papers!

In 2022, a Lawrence Berkeley Laboratory website listed measurements of charger energy. We contacted the website's author, who said those numbers reflect the situation as of 10–20 years ago.[27] He recommended not relying on this data, but to instead collect the data ourselves (Figure 1).

In terms of ML on smartphones, Almeida et al.[2] study the prevalence of ML models and their inference efficiency in Android apps and found the number of models doubled within 12 months with the vast majority being vision-related models. The methodology cannot measure how frequently these models are invoked, but the authors predict that vision model energy consumption has the potential to be significant.

Cai et al.[8] focus on measuring the efficiency of on-device training processes by benchmarking training time, memory use, and energy consumption with on-device models of varying size across different phones and ML libraries. They find that tuning training performance remains complex due to device heterogeneity, asymmetric multiprocessing, and variable batch size. They also report that current on-device ML libraries are optimized primarily for inference, resulting in a significantly larger gap between inference and training efficiency than in the datacenter.

Qui et al.[32] report on a controlled experiment on IoT hardware (NVIDIA Tegra X2 and Xavier NX) that compares energy consumption and estimated carbon emissions in centralized and different federated training settings in the Flower framework. They show a wide range of emissions from federated learning from slightly less than 1 (better than centralized) to ~100 times worse in carbon emissions. These experiments do not appear to include charging efficiency for mobile devices. While they show the variance in average carbon intensity between China, France, and the U.S., they ignore the gains of training at the greenest locations within a country, which can reduce emissions by factors of 5 to 10.[28]

The success and popularity of ML rightfully raised concerns about its environmental impact. Some estimates of CO_2e from ML were so alarming … we wanted to investigate to see what could be done to help. …To our surprise, we found some ML papers overestimated CO_2e by >100,000×.

Given the numerous publications it's diffuclt to be certain, but we believe this is the first article to look holistically at the energy consumption and carbon footprint of ML in smartphones. While other work shows how to calculate power for one charger,[12] this article may also be the first to include the impact of chargers to assess overall smartphone energy use and carbon emissions.

We are certainly not the first to suggest that embodied CO_2e swamps operational CO_2e in information technology.[16,17] It is difficult to calculate the collective impact of new computers accurately. Ideally, embodied CO_2e would be routinely published—like clock rate and thermal design power—for new computing equipment, particularly for servers whose configurations vary more widely than smartphones and PCs.

Conclusion

As Google is a major player in ML, the cloud, *and* smartphones, it is an ideal case study to judge CO_2e from ML. We found that ML energy consumption for cloud datacenters is 10%–15% for 2019–2021[28] and below 3% for smartphones in 2021. Part of the reason for this low level is that modern servers and smartphones include custom hardware to accelerate ML, which also reduces the energy consumption of ML. Indeed, our estimate for ML in datacenters is they consume 70%–80% of the floating-point computation but only 10%–15% of the energy.[28] Presumably, without accelerators, ML could consume 70%-

80% of the energy as well.

We uncovered some surprises related to smartphones. First, most phones have multiple chargers. Our estimates of the mix of wireless and wired chargers and the number of chargers per smartphone suggest that two-thirds of the overall energy smartphones are responsible for is due to their chargers rather than the phones themselves. If end users cut back on the number of chargers or unplug the ones they are not using, we could reduce CO_2e related to smartphones. Second, wireless chargers are much less energy efficient (higher CPUE) than wired chargers. As wireless chargers are growing in popularity, we encourage techniques and policies to mitigate their energy impact. Third, charger prices are not correlated with energy efficiency. Apple and Google show that one can build power efficient chargers at reasonable prices; their chargers cost 20%–70% less than the high-priced ones yet consume 40%–60% less energy.

Another surprise is that on-device computation including training has a built-in disadvantage of up to 25× versus the datacenter. This gap is in part due to its worse PUE and because the cloud allows customers to pick remote datacenters with the cleanest energy nearly anywhere while smartphones must rely on local energy wherever they happen to be. While smartphones can shift charging to non-peak houses, in the cloud we can shift when and where many workloads run. Moreover, one case study suggests training on devices also uses ~12 more energy than in the datacenter. Relying on the cloud and on device for inference and on the cloud for training while following best practices[28] may be the best path to delivering on the amazing potential of ML sustainably.

The high cost of on-device ML training raises an interesting policy issue about the environment and privacy. It's clearly easier to preserve privacy when computation is limited to the edge, but energy consumption and carbon emissions can be two orders of magnitude higher than in the cloud. We believe researchers, society, and policymakers should start a conversation about the relative importance of privacy versus climate change and whether one can dramatically reduce CO_2e on the edge with its inherent privacy protection or to try to improve privacy in the cloud with its inherent environmental advantage to match that of on device computation.

Finally, it's important to get the CO_2e numbers right to ensure we work on the most significant environmental problems in IT (Figure 3). For colleagues interested in global climate change, a prominent target is the embodied costs of computers.

Acknowledgments

Thanks to Jeff Dean, Urs Hölzle, Raj Singh, Brock Taute, and Dennis Yee for feedback. Yun Liu, Stefan Dierauf, and Brett McLarnon helped with federated learning data, and Luis-Miquel Munguia ported it to TPUs and measured its energy. Dave Cassano helped with the power measurements of the chargers. Finally, we thank Tammo Spalink for suggesting we tackle ML on smartphones. We would also like to thank the anonymous reviewers for the remarkable time spent reviewing this article.

References

1. Ahmad, R. et al. A survey on energy estimation and power modeling schemes for smartphone applications. *Int. J. Commun. Syst. 30*, 11 (2017).
2. Almeida, M. et al. Smart at what cost? Characterising mobile deep neural networks in the wild. In *Proceedings of the 21st ACM Internet Measurement Conf.*, 2021, 658–672.
3. Alphabet. *CDP Climate Change Response*, Aug. 2022.
4. Apple. *Use Clean Energy Charging on your iPhone*, June 2023.
5. Bankmycell. *How Many Smartphones Are In The World?* 2022.
6. Berger, D. et al. Reducing embodied carbon is important. *Computer Architecture Today*, 2023.
7. BusinessWire. *One Billion Smartphones Worldwide Have Wireless Charging*. 2021.
8. Cai, D. et al. Towards ubiquitous learning: A first measurement of on-device training performance. In *Proceedings of the 5th Intern. Workshop on Embedded and Mobile Deep Learning*, 2021, 31–36.
9. Carroll, A. and Heiser, G. An analysis of power consumption in a smartphone. *USENIX Annual Tech. Conf.*, 2010, 1–14.
10. Department of Energy. *Compliance Certification Database. Energy Efficiency and Renewable Energy Appliance and Equipment Standards Program*. 2020.
11. Department of Energy. *Code of Federal Regulations, Title 10, 430, Subpart B, Appendix Y: Uniform Test Method for Measuring the Energy Consumption of Battery Chargers*. 2022.
12. EPA. *Greenhouse Gases Equivalencies Calculator - Calculations and References*. 2022.
13. Ferreira, D., Dey, A.K. and Kostakos, V. Understanding human-smartphone concerns: A study of battery life. In *Proceedings of the 2011 Intern. Conf. Pervasive Computing*, 19–33.
14. Fletcher, C. *Private Communication*, May 2023.
15. Google Environmental Report 2022; https://bit.ly/3QAVDMs
16. Gupta, U. et al. Chasing carbon: The elusive environmental footprint of computing. *IEEE Micro 42*, 4 (2022), 37–47.
17. Gupta, U. et al. ACT: Designing sustainable computer systems with an architectural carbon modeling tool. In *Proceedings of the 49th Annual Intern. Symp. Computer Architecture*. June 2022, 784–799.
18. Hoque, M.A. et al. Modeling, profiling, and debugging the energy consumption of mobile devices. *ACM Computing Surveys 48*, 3 (2015), 1–40.
19. HPE product carbon footprint—HPE ProLiant DL345 Gen10 Plus server data sheet; https://bit.ly/3QCyM31.
20. International Energy Agency. *Global Energy & CO2 Status Report 2019, Report Extract Emissions*.
21. Jang, J.W. et al. Sparsity-aware and re-configurable NPU architecture for Samsung flagship mobile SOC. In *Proceedings of the ACM/IEEE 48th Annual Intern. Symp. Computer Architecture*. IEEE, 2021, 15–28.
22. Javed, A., Shahid, M.A., Sharif, M. and Yasmin, M. Energy consumption in mobile phones. *Intern. J. Computer Network & Info. Security 9*, 12 (2017), 18–28.
23. Jouppi, N.P. et al. Ten lessons from three generations shaped Google's TPUv4i. In *Proceedings of the ACM/IEEE 48th Annual Intern. Symp. Computer Architecture*. IEEE, 2021, 1–14.
24. Koningstein, R. *We Now Do More Computing Where There's Cleaner Energy*, May 2021.
25. Lövehagen, N., Malmodin, J., Bergmark, P. and Matinfar, S. Assessing embodied carbon emissions of communication user devices by combining approaches. *Renewable and Sustainable Energy Reviews 183* (2023), 113422.
26. Moen, O.M. et al. Screening life cycle assessment of a new datacenter in Trondheim. *SINTEF Report*, 2022.
27. Meier, A. *Personal Communication*, June 25, 2022.
28. Patterson, D. et al. The carbon footprint of machine learning training will plateau, then shrink. *IEEE Computer 55*, 7 (2022).
29. Perrucci, G.P., Fitzek, F.H. and Widmer, J. Survey on energy consumption entities on the smartphone platform. In *Proceedings of the 2011 IEEE 73rd Vehicular Technology Conf.*, 1–6.
30. Popa, R. *Private Communication*, May 2023.
31. Pramanik, P.K.D. et al. Power consumption analysis, measurement, management, and issues: A state-of-the-art review of smartphone battery and energy usage. *IEEE Access 7* (2019), 182113–182172.
32. Qiu, X. et al. *A First Look Into the Carbon Footprint of Federated Learning*. 2021; arXiv:2102.07627
33. Saraev, A., Gama, M., Piontek, F. and Negi, P. *LCA of Dell Servers R6515, R7515, R6525, R7525*. 2021.
34. Schwartz, R., Dodge, J., Smith, N.A. and Etzioni, O. Green AI. *Commun. ACM 63*, 12 (Dec. 2020), 54–63.
35. Sun, L., Sheshadri, R., Zheng, W. and Koutsonikolas, D. Modeling WiFi active power/energy consumption in smartphones. In *Proceedings of the IEEE 34th Intern. Conf. Distributed Computing Systems*, 2014, 41–51; 10.1109/ICDCS.2014.13.
36. Taiwan Semiconductor Manufacturing Company. *TSMC 2022 Sustainability Report*. 2022
37. Thompson, N.C., Greenewald, K., Lee, K. and Manso, G.F. Deep learning's diminishing returns: The cost of improvement is becoming unsustainable. *IEEE Spectrum 58*, 10 (2021), 50–55.
38. Walker, A. *We Asked, You Told Us: Most Readers Own Plenty of Smartphone Chargers*. 2020.

David Patterson is a Distinguished Engineer at Google in Mountain View, CA, USA, and a Professor Emeritus at the University of California, Berkeley, CA, USA.

Jeffrey M. Gilbert is Director and principal software engineer at Google in Cambridge, MA, USA.

Marco Gruteser is a research scientist at Google in Mountain View, CA, USA.

Efren Robles is Manager, Technical Program Management - AI/ML HW at Google in Mountain View, CA, USA.

Krishna Sekar is a systems architect at Google in Mountain View, CA, USA.

Yong Wei is a data scientist at Google in Mountain View, CA, USA.

Tenghui Zhu is a software engineer at Google in Mountain View, CA, USA.

research highlights

DOI:10.1145/3631338

To view the accompanying paper, visit doi.acm.org/10.1145/3611094

Technical Perspective
How Easy Is It to Describe Hard Polynomials?

By Nitin Saxena

WE STUDY REAL-LIFE problems via mathematical models, which usually tend to be nonlinear systems, or equations. A standard trick is to "linearize" this nonlinear system. The newly obtained system then becomes a subject of linear algebra; at which point we get the happy feeling that we have solved the original nonlinear system. A key object in this path to happiness is the determinant. Practitioners in science and engineering use the determinant to solve a linear system, which is the bedrock of algorithms in optimization, data analysis, graphics, game theory, economics, and business.

The following paper studies a related polynomial that is easier to describe: Take d square matrices, each of order n, multiply them, and consider the top-left entry of the product matrix; this defines the iterated matrix multiplication (IMM) polynomial in dn^2 variables. How compactly can the polynomial IMM be fabricated on a machine? A positive answer to this question would give us practical algorithms, while a negative answer would mean we have identified a hard polynomial that is easy to describe.

If we fabricate term by term, a naïve count says there are n^d terms in IMM. This is exponential in d and becomes impossibly large as the input parameter d grows, say, beyond 100. We want to optimize this situation or prove that no further optimization is possible to fabricate IMM on machine models. The computer science area that studies such questions is *algebraic complexity*, and it is a rapidly growing subarea.

Algebraic complexity develops ideas that are useful in two ways: Upper bound—there are tools to solve problems like computing the determinant and IMM very fast, finding the root of a polynomial system, finding a polynomial factor, and testing whether two polynomials are the same. Lower bound—there are methods to prove that certain natural looking polynomials are hard to fabricate and compute.

Why should a practitioner care about hard polynomials? Shouldn't our focus, as computer scientists, be only on solving problems; say, by adding interesting simplifying assumptions? While that may be our ideal job, there are many problems out there that seem to possess an inherent intractability so that even practical assumptions do not help to solve them in any meaningful way. Surprisingly, such problems may be a boon to other areas; for example, cryptography turns hard problems into practical protocols secure to adversarial attacks. This is what secures the Internet. Thus, it motivates us to find not only an upper bound whenever possible, but also a lower bound proof.

A famous example is the problem of counting the number of perfect matchings in a graph. For instance, you may want to assign tasks to servers in a stable way, given the preference list of each party involved. How many such assignments are possible? This counting problem lives at the heart of algebraic complexity theory and exactly defines a polynomial called permanent, per(M), given a square matrix M. So, now we have brought on stage two polynomials: IMM and per(M). It is widely conjectured that per(M) is the more difficult one among the two; proving its hardness is the central open question in algebraic complexity (called the VP≠VNP question).

The following paper achieves a landmark in the larger quest of understanding hardness, identity testing, and reconstruction.

In contrast, it is not difficult to see that IMM is friendlier, in the sense that a clever circuit can express IMM in a compact way. However, it was an open question to understand the depth of this circuit. In the spirit of parallel processing, the circuit depth signifies the time taken to compute the polynomial; while the size refers to the space or the number of arithmetic processors required to fabricate the polynomial. The paper proves that if we restrict to constant-depth then IMM requires a circuit of very large size. As such, constant-time matrix multiplication requires exponentially many arithmetic processors.

Not only is this lower-bound statement prized, more interesting is the proof method that it develops. It has two important lessons. First, the setting where d (= number of matrices) is significantly smaller than n (= order of each matrix) is amenable to better structural transformations. This allows us to make the circuit multiplication gates very well-behaved (namely, set-multilinear). Second, there is a low-rank matrix associated with these multiplication gates (namely, the partial-derivative matrix). This shows us that IMM has a low-rank matrix, which yields a contradiction, unless the circuit size is superpolynomial.

In the past decades, algebraic complexity has taken big strides in the development of techniques. This paper achieves a landmark in the larger quest of understanding hardness, identity testing, and reconstruction. It encourages us to try newer circuit transformations that linearize a circuit, just enough, to apply linear algebra.

Nitin Saxena is the N. Rama Rao Chair Professor in the Department of Computer Science and Engineering at the Indian Institute of Technology Kanpur, India. Follow his work at https://bit.ly/40w8liJ.

DOI:10.1145/3611094

To view the accompanying Technical Perspective, visit doi.acm.org/10.1145/3631338

Superpolynomial Lower Bounds Against Low-Depth Algebraic Circuits

By Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas

Abstract

An Algebraic Circuit for a multivariate polynomial P is a computational model for constructing the polynomial P using only additions and multiplications. It is a *syntactic* model of computation, as opposed to the Boolean Circuit model, and hence lower bounds for this model are widely expected to be easier to prove than lower bounds for Boolean circuits. Despite this, we do not have superpolynomial lower bounds against general algebraic circuits of depth 3 (except over constant-sized finite fields) and depth 4 (over any field other than $\mathbb{F}_2$), while constant-depth Boolean circuit lower bounds have been known since the early 1980s.

In this paper, we prove the *first superpolynomial lower bounds against algebraic circuits of all constant depths* over all fields of characteristic 0. We also observe that our super-polynomial lower bound for constant-depth circuits implies the first deterministic sub-exponential time algorithm for solving the Polynomial Identity Testing (PIT) problem for all small-depth circuits using the known connection between algebraic hardness and randomness.

1. INTRODUCTION

1.1. Algebraic complexity

The central questions of Complexity theory, such as P vs. NP, have proven to be difficult because it is hard to understand and capture the power of general algorithms. As a result, there have been many distinct lines of research focusing on restricted kinds of algorithms. Quite often, these capture all the approaches we have for solving certain families of computational problems, and thus are interesting objects of study in their own right.

This work is in the area of *Algebraic Complexity*, which is one such line of inquiry that has received a considerable amount of attention in the last few decades (for example, see the book[4] or the surveys.[13,19,20,22] Here, we study algebraic problems that are closely related to multivariate polynomials over some fields $\mathbb{F}$ which we will assume to be the complex numbers $\mathbb{C}$ in this paper. An example of such a polynomial in 3 variables of degree 3 is

$$P(x_1, x_2, x_3) = x_1 + x_2 + x_3 + x_1x_2 + x_2x_3 + x_1x_3 + x_1x_2x_3. \quad (1)$$

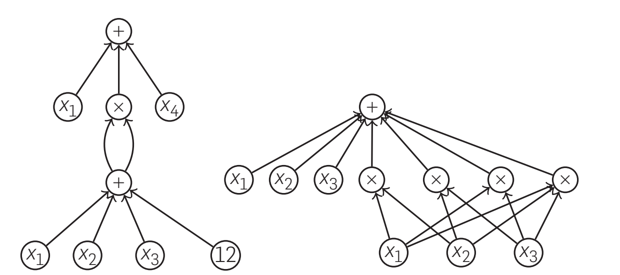
Many natural algorithmic problems can be stated as the problem of evaluating one or more multivariate polynomials at an input point. Important examples of such problems include the Fast Fourier Transform, the Determinant and Matrix Multiplication. Some problems of this form are also expected to be computationally intractable, such as the Permanent.[25] Given such a computational problem, it is natural to consider algorithms that are themselves somewhat algebraic, and use only natural operations such as additions and multiplications. This brings us to the *Algebraic Circuit* model, which is defined by a sequence of instructions, each involving sums (or more generally linear combinations) and products, at the end of which we produce the polynomial P we are interested in evaluating. An algebraic circuit for P gives us a recipe to evaluate P at any given point.[a]

An algebraic circuit is typically visualized as a directed acyclic graph as in Figure 1 (left) where the intermediate nodes (i.e., gates) represent the basic instructions. The efficiency of the computation is captured by the *size* of the circuit, which is defined to be the number of edges in the underlying graph (one could also look at the number of gates, which is within a quadratic factor of the number of edges). Another parameter of efficiency is the *depth* of the circuit, which is the length of the longest path in the computation. The depth captures the extent to which the algorithm is parallelizable. When the underlying graph is a tree, the circuit is said to be a *formula*.[b]

a One can also consider divisions, but they can typically be eliminated without much extra cost.

b We typically consider an infinite sequence of polynomials, a distinction that we frequently gloss over in this paper. We also require that the polynomials have only moderate degree, that is, that the degree grows at most as a polynomial function in the number of input variables. A good example to keep in mind is the problem of computing the determinant of an $n \times n$ matrix. This gives, for each n, a multivariate polynomial of degree n in $N = n^2$ variables.

Figure 1. Left: An algebraic circuit. Right: A depth-2 algebraic circuit suggested by Equation (1).



The original version of this paper was published in the *Proceedings of the 2021 IEEE 62nd Annual Symposium on Foundations of Computer Science,* Denver, CO, USA.

Given a computational problem specified by a polynomial P, we would like to study the smallest size of an algebraic circuit (or formula) for P. In particular, there are many situations where we would like to show that a given polynomial has no small algebraic circuits. Analogous to the P vs. NP question in standard complexity theory, the principal question of this area is the Valiant's P vs. Valiant's NP (VP vs. VNP) question, which can equivalently be stated as the following "lower bound" question: show that the Permanent polynomial, which is a polynomial in $N = n^2$ variables of degree n has no algebraic circuits of size poly(N).

As the algebraic circuit model is required to construct the formal polynomial P under consideration, it is a *syntactic* model of computation, as opposed to the Boolean circuit model, which is typically only required to output the correct answer on certain inputs (e.g., Boolean inputs). As a consequence, it is intuitive that the algebraic circuit yields a more restricted family of algorithms than its Boolean variant. In particular, this implies that the problem of proving algebraic circuit lower bounds is easier than its Boolean counterpart.[3] On the other hand, it is also a natural model that captures most algorithms for algebraic computational problems of the kind we care about. Resolving the VP vs. VNP question is thus seen as an important stepping stone to resolving P vs. NP.

1.2. Constant-depth circuits

What makes the VP vs. VNP question (and other similar questions in algebraic complexity) particularly alluring is that there is a very concrete strategy for resolving it based on the technique of *depth-reduction*. To describe this, let us start with some basic notation. A polynomial P can always be written as a sum of monomials as in Equation (1) above. Such a representation can also be visualized as a depth-2 circuit (see Figure 1 right) and is called a $\Sigma\Pi$ circuit for the reason that the polynomial is expressed as a sum of products of variables. Such a representation is extremely simple to analyze: for example, any polynomial P has a unique such minimal representation, the size of which is essentially the number of monomials in the polynomial. For a polynomial in N variables of degree d, this is typically around N^d.

However, things get much more interesting with just one more layer of complexity. For instance, consider $\Sigma\Pi\Sigma$ circuits, which are sums of products of sums of variables. Such circuits can be much more succinct than $\Sigma\Pi$. For instance, the polynomial from Equation (1) can be written more succinctly as

$$(1+x_1)(1+x_2)(1+x_3)-1,$$

which can be seen as a simple $\Sigma\Pi\Sigma$ circuit. More generally, a surprising and beautiful result of Gupta et al.[9] (building on many previous results[1,12,24]) showed that if a polynomial P on N variables of degree d has an algebraic circuit of size poly(N), then it has a $\Sigma\Pi\Sigma$ circuit of size $N^{O(\sqrt{d})}$. While this is quite large, it is still considerably smaller than the trivial N^d in the $\Sigma\Pi$ case. In particular, it is feasible that we could show VP is not VNP simply by showing that the Permanent (or another similar polynomial) has no $\Sigma\Pi\Sigma$ circuits of this size!

In terms of depth, $\Sigma\Pi\Sigma$ circuits are just depth-3 circuits. So this means that a strong enough lower bound for depth-3 circuits implies a separation between VP and VNP. No such connection between constant-depth circuit lower bounds and general circuit lower bounds is known in the non-algebraic (i.e., Boolean) setting.

In particular, the results above illustrate how useful it would be to have strong constant-depth circuit lower bounds in the algebraic setting. Unfortunately, however, these bounds have so far been difficult to demonstrate. Despite extensive work in the area, and many beautiful lower bound results against restricted models of circuits,[8,15,16] even super-polynomial lower bounds against $\Sigma\Pi\Sigma$ circuits have remained an open question.

In this work, we show superpolynomial lower bounds against general $\Sigma\Pi\Sigma$ circuits and more generally, against circuits whose depth is a constant independent of the number of variables and the degree of the underlying polynomial. More precisely we show the following.

Theorem 1 (Main Result). *Let N, d be growing parameters with $d = o(\log N)$. There is an explicit polynomial $P_{N,d}(x_1, \ldots, x_N)$ that has no algebraic circuits of constant depth and polynomial size. More precisely, for any fixed Δ, there is a constant $\varepsilon > 0$ such that any circuit of depth Δ computing $P_{N,d}$ has size at least $N^{\Omega(d^{\varepsilon})}$.*

In the case of $\Sigma\Pi\Sigma$ circuits (i.e., $\Delta = 3$), we get a lower bound of $N^{\Omega(\sqrt{d})}$. By the result of Gupta et al.[9] mentioned above, any asymptotic improvement in the exponent in this bound could potentially separate VP from VNP.

Moreover, the polynomial $P_{N,d}$ itself is easy to describe and encodes the natural problem of Matrix Multiplication. Assume n and d are such that $N = dn^2$. The polynomial $P_{N,d}$ is the Iterated Matrix Multiplication polynomial $\mathrm{IMM}_{n,d}$ on $N = dn^2$ variables, defined as follows. The underlying variables are partitioned into d sets $X_1, \ldots, X_d$ of size n^2, each of which is visualized as an $n \times n$ matrix with distinct variable entries. Then $\mathrm{IMM}_{n,d}$ is defined to be the polynomial that is the (1, 1)th entry of the product matrix $X_1 \cdot X_2 \cdots X_d$.

1.3. Consequence of polynomial identity testing

Another important question in algebraic complexity deals with the algorithmic analysis of a given algebraic circuit. More specifically, the Polynomial Identity Testing (PIT) problem is the following algorithmic problem: given an algebraic circuit C computing a polynomial P, is this polynomial a nonzero polynomial? That is, are any of the coefficients of P non-zero?

To illustrate the hardness of this question, it is worth noting that the number of potential monomials in the polynomial P is *exponentially large*, and hence expanding out to compute each of the coefficients of P would take exponential time. Nevertheless, it is known that there is a polynomial-time *randomized* algorithm for this problem. We simply evaluate P at a random point and check if P evaluates to a non-zero value at that point. The circuit representation allows for efficient evaluation of P and hence this algorithm is efficient. However, the algorithm might make an error: specifically, this happens when P is non-zero but the algorithm happens to choose a point where P evaluates to zero. However, the probability of this can be shown to be small.

The algorithmic challenge is therefore to come up with a *deterministic* (i.e., non-randomized) efficient algorithm for this polynomial. This problem has seen extensive work over the past couple of decades (see e.g., the surveys[13,20,22]), with the introduction of many specialized tools to solve restricted variants of this problem. Nevertheless, even sub-exponential time deterministic algorithms for PIT for simple classes such as $\Sigma\Pi\Sigma$ circuits remained open until this work.

The difficulty of this problem is explained by the strong connections that have been made between this question and the problem of showing lower bounds against algebraic circuits. An influential result of Kabanets and Impagliazzo[11] showed that superpolynomial lower bounds against general algebraic circuits imply deterministic *sub-exponential* time algorithms for general PIT. Later results have tried to extend this *algebraic hardness vs. randomness* framework in several different ways.[5,6,13] Specifically, Dvir et al.[6] proved that hardness for constant-depth circuits implies a deterministic algorithm for PIT for constant-depth circuits. In a recent paper, Chou et al.[5] refined this result and improved the dependence on the degree of the polynomial.

We observe that this result from Chou et al.[5] combined with our lower bound from Theorem 1 gives the first sub-exponential time deterministic algorithm for PIT for algebraic circuits of any constant depth. Specifically, we get the following.

Corollary 2. *The following holds for any $\varepsilon > 0$. Let C be an algebraic circuit of size $s \leq \mathrm{poly}(n)$ and constant depth Δ computing a polynomial on n variables. There is a deterministic algorithm that can check whether the polynomial computed by C is identically zero or not in time $(s^{\Delta+1} \cdot n)^{O(n^{\varepsilon})}$.*

2. THE APPROACH: "HARDNESS ESCALATION"

While lower bounds against general algebraic circuits have been hard to prove, we do have several beautiful results against restricted kinds of algebraic circuits, such as *Homogeneous, Multilinear,* and *Set Multilinear* circuits. As these will be useful in the sequel, we review some of these definitions below.[c]

Recall that a multilinear polynomial $P(x_1, \ldots, x_N)$ is one in which each variable x_i has degree at most 1, and a homogeneous polynomial is one that is a linear combination of monomials of the same total degree. If the underlying variable set is partitioned into d variable sets $X_1, \ldots, X_d$, then P is said to be *set-multilinear* with respect to this variable partition if P is a linear combination of monomials that contains one variable from each variable set among $X_1, \ldots, X_d$; note that a set-multilinear polynomial is both multilinear and homogeneous (of degree d). For example, the $n \times n$ Determinant is a set-multilinear polynomial w.r.t the variable partition corresponding to the rows of the underlying matrix, and the polynomial $\mathrm{IMM}_{n,d}$ defined above is set-multilinear w.r.t. the partition into matrices $X_1, \ldots, X_d$.

Given a set-multilinear polynomial P (w.r.t. variable partition $X_1, \ldots, X_d$), it is natural to look at algebraic circuits computing P that themselves have the same structure. In particular, an algebraic circuit is said to be set-multilinear if each internal gate computes a set-multilinear polynomial in a subset of $X_1, \ldots, X_d$. Similarly, a multilinear or homogeneous circuit is one where each internal node computes a multilinear or homogeneous polynomial respectively. For each such restricted type of circuit, we have nontrivial lower bounds on the sizes of circuits computing explicit polynomials (also restricted in the same way).[14,15,16] An important result of Nisan and Wigderson[15] proved lower bounds against small-depth set-multilinear and homogeneous circuits computing $\mathrm{IMM}_{n,d}$. Building upon this, Raz[16] showed superpolynomial lower bounds on the size of any (unbounded depth) multilinear *formula* computing the $n \times n$ Determinant and Permanent.

It is natural to ask if we can use these lower bounds against restricted kinds of circuits to prove lower bounds against more general algebraic circuits. Such "hardness escalation" results have appeared in many areas in computational complexity, including Algebraic complexity theory. Strassen[23] and Raz[17] both observed (in different settings) that lower bounds against small-depth circuits computing low-degree polynomials imply lower bounds against larger depth circuits. More recently, Raz[18] showed that if a homogeneous or set-multilinear polynomial of degree d has an algebraic formula of size s, then it also has a *homogeneous* or *set-multilinear* formula of size $\mathrm{poly}(s) \cdot (\log s)^{O(d)}$ respectively. In particular, for a homogeneous (resp. set-multilinear) polynomial P of degree $d = O(\log N / \log\log N)$, it follows that P has a formula of size $\mathrm{poly}(N)$ if and only if P has a homogeneous (resp. set-multilinear) formula of size $\mathrm{poly}(N)$.[d]

The latter result implies that if we could prove homogeneous or set-multilinear formula lower bounds of the form $N^{\omega_d(1)}$ (i.e., the exponent goes to infinity with d) for a polynomial P with N variables and degree d, then we would have superpolynomial general algebraic formula lower bounds. In particular, this would imply lower bounds against constant-depth algebraic circuits, as any constant-depth algebraic circuit can be converted to an algebraic formula with only a polynomial blow-up.[e]

Unfortunately, known results do not yield such lower bounds. In the homogeneous case, we have strong lower bounds against certain formulas of depth at most 4,[14,15] but this falls short of proving anything for general formulas as Raz's "homogenization" result does not preserve the depth of the formula (in fact, known results for homogeneous formulas stop yielding lower bounds exactly in the regime where they would yield implications for general circuits). In the set-multilinear, and more generally multilinear case, we do have lower bounds against formulas of large depth,[15,16] but all such lower bounds are of the form $f(d) \cdot \mathrm{poly}(N)$ where $f(d)$ is a superpolynomial (and subexponential) function of d. With analogy to *Parameterized Complexity Theory*, we call such bounds *Fixed Parameter*

c The random point is typically picked uniformly at random from a large finite grid of points.

d This terminology appeared in a result of Beame et al.[2] on proof complexity. The authors of that paper attribute the term to Rahul Santhanam.

e Raz's result is slightly stronger for homogeneous formulas, but we ignore this point here.

Tractable (FPT) bounds. Our motivating question is if we can prove strong *non-FPT lower bounds* against restricted types of circuits or formulas in a setting where we can use them for lower bounds for general algebraic circuits or formulas. We show that this is indeed possible.

2.1. Technical results

Our main lower bound result is a strong non-FPT lower bound against constant-depth set-multilinear circuits, considerably strengthening known results in this direction.

We prove our lower bounds for the $\mathrm{IMM}_{n,d}$ polynomial on $N = dn^2$ variables as defined above. This polynomial has a simple divide-and-conquer-based set-multilinear formula of size $n^{O(\log d)}$, and more generally for every $\Delta \leq \log d$, a set-multilinear formula of depth 2Δ and size $n^{O(\Delta d^{1/\Delta})}$. It is reasonable to conjecture that this simple upper bound is tight up to the constant in the exponent for set-multilinear formulas.

This has been shown for set-multilinear, and more generally, for homogeneous circuits of depths 3 and 4.[7,14,15] However, as far as we know, no superpolynomial non-FPT lower bounds were known for any depths greater than 4, even under the set-multilinearity restriction. We show such lower bounds for all constant depths.

Theorem 3 (Lower bound against set-multilinear circuits). *Assume* $d \leq (\log n)/100$. *For any constant depth* Δ, *any set-multilinear circuit C computing* $\mathrm{IMM}_{n,d}$ *of depth at most* Δ *must have size at least* $n^{d^{\varepsilon}}$, *where* ε *is a constant that depends only on* Δ. *In the particular case that* $\Delta = 5$, *the size of C must be at least* $n^{\Omega(\sqrt{d})}$.

With these stronger non-FPT lower bounds for set-multilinear circuits in place, we are able to derive lower bounds against stronger families of algebraic circuits via hardness escalation arguments.

First, we show that any homogeneous circuit of depth Δ and size s computing a set-multilinear polynomial P of degree d can be converted to a set-multilinear circuit *with the same depth* for P of size $s \cdot d^{O(d)}$. Putting this together with Theorem 3, we get the first superpolynomial lower bounds (FPT or non-FPT) for homogeneous circuits of depth greater than 5.

Corollary 4 (Lower bound against homogeneous circuits). *Assume* $d \leq (\log n)/100$. *For any constant depth* Δ, *any homogeneous circuit C computing* $\mathrm{IMM}_{n,d}$ *of depth at most* Δ *must have size at least* $n^{d^{\varepsilon}}$, *where* ε *is a constant that depends only on* Δ. *In the particular case that* $\Delta = 5$, *the size of C must be at least* $n^{\Omega(\sqrt{d})}$.

Note that our improved non-FPT bounds are crucial for deriving the above result from Theorem 3. The previous best lower bound of $2^{\Omega(\sqrt{d})}$ due to Nisan and Wigderson[15] does not suffice for this, because of the $d^{O(d)}$ blow-up involved in converting a homogeneous circuit to one that is set-multilinear.

Next, we show that any (possibly non-homogeneous) algebraic circuit of depth Δ and size s computing a homogeneous polynomial P of degree d can be converted to a homogeneous circuit for P of depth $2\Delta + 1$ and size $\mathrm{poly}(s) \cdot 2^{O(\sqrt{d})}$. This implies the main result Theorem 1.

3. AN ILLUSTRATION: DEPTH-3 CIRCUITS

We prove in this section the case $\Delta = 5$ of Theorem 3 and Corollary 4 and the case $\Delta = 3$ of Theorem 1.

3.1. Reduction to set-multilinear depth 5

In the next sub-section, we will show super polynomial lower bounds for small-degree polynomials against set-multilinear formulas of some constant depth. We want to extend these lower bounds to the general setting (i.e., without the set-multilinearity constraint).

Raz[18] showed that if there is a fanin-2 formula of size s and depth Δ that computes a set-multilinear polynomial over the disjoint sets $(X_1, \ldots, X_d)$, then there exists also a fanin-2 set-multilinear formula of size $O((\Delta+2)^d s)$ and depth Δ that computes the same polynomial. However, the fanin-2 constraint is an issue when we want to deal with constant-depth circuits.

We show here that we can get a similar result for circuits with arbitrary fanins at the cost of a size blow-up of $d^{O(d)}$ poly(s) and an increase of the depth by a factor of at most 2.

Proposition 5. *Let* s, N, d *be growing parameters with* $s \geq Nd$. *If C is a circuit of size at most s and depth at most* 3 *computing a set-multilinear polynomial P over the sets of variables* $(X_1, \ldots, X_d)$ *(with* $|X_i| \leq N$*), then there is a set-multilinear circuit* $\tilde{C}$ *of size* $d^{O(d)}$poly(s) *and depth at most* 5 *computing P.*

Similar to Raz's approach, we start by *homogenizing* the circuit and then we *set-multilinearize* it. In particular, the previous proposition is just the composition of Lemmas 6 and 7.

Non-homogeneous to homogeneous circuits. It was already noticed[9,10] that we can convert nonhomogeneous formulas of depth 3 to homogeneous formulas of depth 5 with a relatively small size blow-up.

Indeed, a general $\Sigma\Pi\Sigma$ circuit of size s yields a formula of the following kind

$$F = \sum_{i=1}^{s} \prod_{j=1}^{s} \ell_{i,j}$$

where each $\ell_{i,j}$ is an affine linear polynomial in the underlying variables. Note that the individual summands of the expression may compute polynomials of degree s, which is possibly much larger than d. The main observation is that, assuming that the underlying field $\mathbb{F}$ has characteristic 0 (which is the case here since we chose $\mathbb{F} = \mathbb{C}$), the homogeneous degree-d part of each summand can be computed by a homogeneous $\Sigma\Pi\Sigma\Pi\Sigma$ formula of size $\mathrm{poly}(s) \cdot \exp(O(\sqrt{d}))$. Replacing each of these terms with such a formula, we see that the same polynomial can also be computed by a homogeneous $\Sigma\Pi\Sigma\Pi\Sigma$ formula of size $\mathrm{poly}(s) \cdot \exp(O(\sqrt{d}))$.[f]

The main observation is also easy to prove. Consider any summand $T_i = \ell_{i,1} \cdots \ell_{i,s}$. It suffices to prove the observation in the case that each $\ell_{i,j}$ has a non-zero constant term c_j (it is easy to reduce to this case). In this case, we can write

f A formula is called a fanin-2 formula, if every gate has at most 2 input wires.

$$T_i = \left(\prod_{j=1}^{s} c_i\right) \cdot \prod_{j=1}^{s}\left(1+\ell'_{i,j}\right)$$

where each $\ell'_{i,j}$ is a homogeneous linear polynomial. It then follows that the degree-d homogeneous part $T_{i,d}$ of T_i can be written as a linear projection applied to the *Elementary Symmetric Polynomial* E_s^d of degree d in s variables. More precisely, we have

$$T_{i,d} = \left(\prod_{i=1}^{s} c_i\right) \cdot E_s^d\left(\ell'_{i,1},\ldots,\ell'_{i,s}\right).$$

Shpilka and Wigderson[21] [Theorem 5.3] proved that, over fields of characteristic 0 the polynomial E_s^d has a homogeneous $\Sigma\Pi\Sigma\Pi$ circuit of size $\text{poly}(s)\cdot\exp(O(\sqrt{d}))$. The result seems to be also proved independently in Hrubeš and Yehudayoff.[10] Using this with the above expression, we get the following result.

LEMMA 6 (LEMMA 5.6 IN THE JOURNAL VERSION)[9]. *Let s, N, d be growing parameters. Fix any $\Sigma\Pi\Sigma$ circuit F of size at most s computing a homogeneous polynomial $P(x_1, \ldots, x_N)$ of degree d. Then, P can also be computed by a* homogeneous $\Sigma\Pi\Sigma\Pi\Sigma$ *circuit F' of size at most* $\text{poly}(s)\exp(O(\sqrt{d}))$.

Homogeneous to set-multilinear circuits. Then we need to convert homogeneous circuits into set-multilinear ones without increasing the depth and with a relatively small size blow-up. In fact, this transformation is not really more complicated in the case of an arbitrary depth. Thus, we place ourselves here directly in this case.

LEMMA 7. *Let Δ be an odd integer. Let s, N, d be growing parameters with $s \geq Nd$. If C is a homogeneous circuit of size at most s and depth at most Δ computing a set-multilinear polynomial P over the sets of variables $(X_1, \ldots, X_d)$ (with $|X_i| \leq N$), then there is a set-multilinear circuit $\tilde{C}$ of size at most $(d!)s$ and depth at most Δ computing P.*

PROOF. Let us describe our new circuit $\tilde{C}$. For any gate α of degree d_α from C, we create $\binom{d}{d_\alpha}$ gates α_S in $\tilde{C}$ (we index these gates by the subsets $S \subseteq [d]$ such that $|S| = d_\alpha$). Now we want to link these gates such that for every gate α in C and any $S \subseteq [d]$ with $|S| = d_\alpha$, the depth of α_S is the same as the one of α and the polynomial computed by α_S is the projection of the polynomial computed by α to the set-multilinear part associated with S:

$$\alpha_S = \sum_{m \text{ set-multilinear over } (X_i)_{i\in S}} ([m]\alpha)\, m$$

where $([m]\alpha)$ is the coefficient of the monomial m in α.

Let us do it by induction on the structure of the graph.

If α is a leaf, it is labeled either by a constant or by a variable. When $d_\alpha = 0$, there is nothing to change. Otherwise $d_\alpha = 1$. In C the leaf α is labeled by a variable x which belongs to X_i. We just need to label the gates by $\alpha_{\{i\}} = x$ and $\alpha_{\{j\}} = 0$ for $j \neq i$.[g]

If $\alpha = c^1\alpha^1 + \ldots + c^p\alpha^p$ is a sum gate (where the c^i are constants in $\mathbb{C}$), we just need to compute the linear combination part by part. For any $S \subseteq [d]$ with $|S| = d_\alpha$:

$$\alpha_S = c^1\alpha_S^1 + \ldots + c^p\alpha_S^p.$$

Finally, if $\alpha = \alpha^1 \cdot \ldots \cdot \alpha^p$ is a product gate, we need to extract all the decompositions. Let $S \subseteq [d]$ with $|S| = d_\alpha$:

$$\alpha_S = \sum_{\substack{(S_1,\ldots,S_p) \text{ partition of } S \\ \text{with } \forall i, |S_i| = d_{\alpha^i}}} \alpha_{S_1}^1 \cdot \ldots \cdot \alpha_{S_p}^p.$$

The size of the sum is $\binom{d_\alpha}{d_{\alpha^1},\ldots,d_{\alpha^p}}$.

Hence each leaf and sum gate α in C creates $\binom{d}{d_\alpha} \leq d!$ new gates in $\tilde{C}$. Each multiplication gate α in C creates $\binom{d}{d_\alpha} \leq d!$ sum gates and $\binom{d}{d_\alpha}\binom{d_\alpha}{d_{\alpha^1},\ldots,d_{\alpha^p}} \leq d!$ new product gates. So the number of gates of $\tilde{C}$ is bounded by $2d!$ times the number of gates of C. Notice that we can avoid factor 2 since we do not need to keep the sum gates which come from a product gate, we can merge them with the sum gates of the next layer of the circuit. (We always assume that the output gate is a sum gate, and so there is always a next layer to merge with.) Furthermore, the depth of the gate α_S in $\tilde{C}$ is the same as the one of the gate α in C. □

3.2. Lower bounds against set-multilinear depth 5

By Proposition 5, it is sufficient to get a sufficiently large lower bound against set-multilinear depth-5 circuits to prove Theorem 1 for depth-3 circuits.

In this section, we will prove the following non-FPT lower bound against set-multilinear depth-5 circuits.

LEMMA 8. *Let n, $d \in (N)\setminus\{0\}$ with $n \geq 4^{\sqrt{d}+1}$. Any set-multilinear circuit C of depth* 5 *computing* $\text{IMM}_{n,d}$ *has size at least* $n^{\Omega(\sqrt{d})}$.

Notice that if $d \leq (\log n)/100$, then the hypothesis of this lemma is immediately satisfied. Consequently, this lemma directly implies the case $\Delta = 5$ of Theorem 3.

In the case where $\Delta = 3$ in Theorem 1 using Proposition 5, we can transform circuit C into a depth-5 set-multilinear one of size at most $d^{O(d)}\text{poly}(s)$. By Lemma 8, it implies that $d^{O(d)}\text{poly}(s) \geq n^{\Omega(\sqrt{d})}$. By the assumption $d \leq (\log n)/100$, we get the desired lower bound for s.

The case $\Delta = 5$ of Corollary 4 is proved identically by replacing Proposition 5 with Lemma 7.

Therefore, all we have to do is prove Lemma 8. The proof technique is very standard: we will focus on the complexity measure of partial derivatives (introduced by Nisan and Wigderson[15]) and show that particular polynomials computed by small set-multilinear circuits of depth 5 have small measures. The novelty here is that we will focus on polynomials which are *lopsided*, that is, that are based on a partition of the variables where different parts have different sizes. It is thanks to this point that we are able to obtain new lower bounds.

The complexity measure. So let us start by introducing the measure.

We will consider the set of words on an alphabet $A \subseteq \mathbb{N}\setminus\{0\}$. Let $w = (w_1,\ldots,w_d) \in A^d$ For a subset $S \subseteq [d]$, let w_S denote

g In fact, they claim the result for general depth-4 circuits, but it was already noticed in Gupta et al.[9] that the formula they get with this approach is homogeneous. In fact in Gupta et al.[9] they also show that the product gates can be replaced by exponentiation gates, but we do not need it here.

$\sum_{i \in S} w_i$. We say $w \in A^d$ is *b-unbiased* if $|w_{[t]}| \leq b$ for every $t \leq d$. We define $\mathcal{P}_w = \{i \mid w_i > 0\}$ and $\mathcal{N}_w = \{i \mid w_i < 0\}$, that is, the positive and negative indices of w respectively.

Given w, we denote by $\bar{X}(w)$ a tuple of d sets of variables $(X(w_1), \ldots, X(w_d))$ where $|X(w_i)| = 2^{|w_i|}$. We denote by $\mathbb{F}_{\text{sm}}[\mathcal{T}]$ the set of set-multilinear polynomials over the tuple of sets of variables $\mathcal{T}$.

Let $\mathcal{M}_w^{\mathcal{P}}$ and $\mathcal{M}_w^{\mathcal{N}}$ denote the sets of the set-multilinear monomials over only the positive and only the negative variable sets. Let $f \in \mathbb{F}_{\text{sm}}[\bar{X}(w)]$, we define $M_w(f)$ as the matrix of size $|\mathcal{M}_w^{\mathcal{P}}| \times |\mathcal{M}_w^{\mathcal{N}}|$, where the rows are indexed by $\mathcal{M}_w^{\mathcal{P}}$ and the columns by $\mathcal{M}_w^{\mathcal{N}}$ and where the coefficient at the entry (m_1, m_2) is the coefficient of the monomial $m_1 m_2$ in f.

We associate with the space $\mathbb{F}_{\text{sm}}[\bar{X}(w)]$ the standard rank-based complexity measure relrk_w (short for "relative rank") defined as follows. Let $f \in \mathbb{F}_{\text{sm}}[\bar{X}(w)]$ and define

$$\text{relrk}_w(f) = \frac{\text{rank}(M_w(f))}{\sqrt{|\mathcal{M}_w^{\mathcal{P}}| \cdot |\mathcal{M}_w^{\mathcal{N}}|}} = \frac{\text{rank}(M_w(f))}{2^{\frac{1}{2}\sum_{i \in [d]} |w_i|}}.$$

Intuitively, as we will want to play on the size of the subsets of variables and thus modify the maximum rank that the polynomial can have, using the relative rank rather than the usual rank allows us to more easily take into account the distance we are from the maximum rank.

We use the following properties of relrk_w.

Claim 9.

1. *(Imbalance) Say* $f \in \mathbb{F}_{sm}[\bar{X}(w)]$. *Then,* $\text{relrk}_w(f) \leq 2^{-|w_{[d]}|/2}$.
2. *(Sub-additivity) Say* $f, g \in \mathbb{F}_{sm}[\bar{X}(w)]$. *Then* $\text{relrk}_w(f+g) \leq \text{relrk}_w(f) + \text{relrk}_w(g)$.
3. *(Multiplicativity) Say* $f = f_1 \cdot f_2 \cdot \ldots \cdot f_t$ *and assume that for each* $i \in [t], f_i \in \mathbb{F}_{sm}[\bar{X}(w_{|S_i})]$, *where* $(S_1, \ldots, S_t)$ *is a partition of* $[d]$ *and for each* $i \in [t]$, $w_{|S_i}$ *stands for the sub-word of* w *indexed by* S_i. *Then*

$$\text{relrk}_w(f) = \text{relrk}_w(f_1 \cdot f_2 \cdot \ldots \cdot f_t) = \prod_{i \in [t]} \text{relrk}_{w_{|S_i}}(f_i).$$

Proof. We have that $|\mathcal{M}_w^{\mathcal{P}}| = 2^{\sum_{i \in \mathcal{P}_w} w_i}$ and $|\mathcal{M}_w^{\mathcal{N}}| = 2^{-\sum_{i \in \mathcal{N}_w} w_i}$. So $2^{|w_{[d]}|}$ is just the ratio of the larger dimension of $M_w(f)$ by the smaller one. As the rank of a matrix is bounded by the minimum between its number of rows and its number of columns, it implies the first inequality of the claim.

The subadditivity property directly follows from the facts that $M_w(f+g) = M_w(f) + M_w(g)$ and that the rank of a matrix is subadditive.

The multiplicative argument is standard too. As the product is set-multilinear, it implies that the matrix $M_w(f_1 \cdot \ldots \cdot f_t)$ is the matrix $M_w(f_1) \otimes \ldots \otimes M_w(f_t)$ where the symbol $\otimes$ stands for the Kronecker product. Finally, the rank is known to be multiplicative with respect to the Konecker product. So,

$$\text{relrk}_w(f_1 \cdot f_2 \cdot \ldots \cdot f_t) = \frac{\text{rank}(M_w(f_1 \cdot \ldots \cdot f_t))}{2^{\frac{1}{2}\sum_{j \in [d]} |w_j|}} \qquad \square$$

$$= \prod_{i \in [t]} \frac{\text{rank}(M_w(f_i))}{2^{\frac{1}{2}\sum_{j \in S_i} |w_j|}} = \prod_{i \in [t]} \text{relrk}_{w_{|S_i}}(f_i).$$

Word polynomials and iterated matrix multiplication polynomial. We said we will prove lower bounds for lopsided polynomials. Unfortunately, all variables parts of the polynomial $\text{IMM}_{n,d}$ have the same size (which is n^2). In fact, we will prove the lower bound for the word polynomial which is a "set-multilinear" projection of $\text{IMM}_{n,d}$ (The projection is called set-multilinear because it preserves the set-multilinearity). So let us define this polynomial.

Let $w \in A^d$ be any word. For any such word, we define a polynomial P_w. Say $\bar{X}(w) = (X_1, \ldots, X_d)$ and since each X_i has size $2^{|w_i|}$, we assume that the variables of X_i are labeled by strings in $\{0,1\}^{|w_i|}$.

Given any monomial $m \in \mathbb{F}_{\text{sm}}[\bar{X}(w)]$, let m_+ denote the corresponding "positive" monomial from $\mathcal{M}_w^{\mathcal{P}}$ and m_- the corresponding "negative" monomial from $\mathcal{M}_w^{\mathcal{N}}$. As each variable of $\bar{X}(w)$ is labeled by a Boolean string and each set-multilinear monomial over any subset of $\bar{X}(w)$ is associated with a string of variables, we can associate any such monomial m' with a Boolean string $\sigma(m')$. More precisely, if $j_1 < \cdots < j_t$ and $m' = x_{\sigma_1}^{(j_1)} x_{\sigma_2}^{(j_2)} \cdots x_{\sigma_t}^{(j_t)}$ with $x_{\sigma_i}^{(j_i)} \in X_{j_i}$ and $\sigma_i \in \{0,1\}^{|w_{j_i}|}$ for each $i \in [t]$, then $\sigma(m')$ is defined to be $\sigma_1 \cdots \sigma_t$. If w is b-unbiased, the difference of length of the strings $\sigma(m_+)$ and $\sigma(m_-)$ is at most b. We will write $\sigma(m_+) \sim \sigma(m_-)$ when the shorter one is a prefix of the other one.

The polynomial P_w is defined as follows:

$$P_w = \sum_{m \in \mathbb{F}[\bar{X}(w)],\ \sigma(m_+) \sim \sigma(m_-)} m.$$

Clearly, the matrices $M_w(P_w)$ are full-rank (i.e., have a rank equal to either the number of rows or the number of columns, whichever is smaller). So, $\text{relrk}_w(P_w) = 2^{-|w_{[d]}|/2} \geq 2^{-b/2}$.

We observe that P_w can be obtained as a *set-multilinear restriction* of $\text{IMM}_{n,d}$ for an appropriate choice of n. Formally, one can show the following. The proof is straightforward but omitted from this extended abstract.

Lemma 10. *Let* $w \in A^d$ *be any word which is b-unbiased. If there is a set-multilinear circuit computing* $\text{IMM}_{2^b,d}$ *of size s and depth* Δ, *then there is also a set-multilinear circuit of size s and depth* Δ *computing a polynomial* $P_w \in \mathbb{F}_{sm}[\bar{X}(w)]$ *such that* $\text{relrk}_w(P_w) \geq 2^{-b/2}$.

Proof of Lemma 8. We now have all the ingredients to tackle the proof of the lemma.

It is a standard fact that any circuit of constant depth can be converted to a formula with only a polynomial blow-up. So it suffices to show the following.

Claim 11. *Let* $d \geq 16$ *and* $k > 2\sqrt{d}$ *be an integer. Let w be any word of length d on the alphabet* $\{-k, \lfloor k - k/\sqrt{d} \rfloor\}$. *Then any set-multilinear formula C of depth* 5 *and of size s satisfies*

$$\text{relrk}_w(C) \leq s \cdot 2^{-\frac{k\sqrt{d}}{8}}.$$

Indeed, by fixing $k = \lfloor \log_2 n \rfloor$, we have $k > 2\sqrt{d}$. We can construct by induction a word w on the alphabet $\{-k, \lfloor k - k/\sqrt{d} \rfloor\}$ which is k-unbiased. Indeed, if $|w_{[i]}| \leq 0$, we choose $w_{i+1} = \lfloor k - k/\sqrt{d} \rfloor$, otherwise we set $w_{i+1} = -k$. By Lemma 10 and Claim 11, we get the lower bound:

$$s \geq 2^{\frac{k\sqrt{d}}{8}} 2^{-k} \geq 2^{(\log_2 n - 1)\frac{\sqrt{d}}{8} - \log_2 n}$$

$$\geq n^{\frac{\sqrt{d}}{8}} 2^{-\frac{\log_2 n}{16} - \log_2 n} \geq n^{\frac{\sqrt{d}}{8} - \frac{17}{16}}$$

for the polynomial $\mathrm{IMM}_{2^k,d}$ against set-multilinear circuits of depth 5.

Proof of Claim 11. We know C is a depth 5 formula, so we can define $C = C_1 + \ldots + C_t$ where each C_i is of the form $\Pi\Sigma\Pi\Sigma$ and has size s_i. We say that C_i is of type 1 if some factor of C_i has degree $\geq \sqrt{d}/2$, otherwise it is of type 2.

- If C_i is of type 1, then $C_i = C_{i,1} \cdot \ldots \cdot C_{i,t_i}$. Up to reordering, we can assume that $C_{i,1}$ is a sum of products of linear forms of degree at least $\sqrt{d}/2$. Notice that if L is a linear form on variables $X(w_i)$, we have $\mathrm{relrk}(L) \leq 2^{-|w_i|/2} \leq 2^{-(k-k/\sqrt{d}-1)/2}$. In particular, by the multiplicativity and sub-additivity of relrk_w (Claim 9),

$$\mathrm{relrk}_w(C_i) \leq \mathrm{relrk}_w(C_{i,1}) \leq s_i 2^{-\frac{k\sqrt{d}-k-\sqrt{d}}{2\sqrt{d}}\deg(C_{i,1})}$$

$$\leq s_i 2^{-\frac{k\sqrt{d}-k-\sqrt{d}}{4}} \leq s_i 2^{-\frac{k\sqrt{d}}{8}}.$$

- If C_i is of type 2, then $C_i = C_{i,1} \cdot \ldots \cdot C_{i,t_i}$ where each factor C_{ij} has degree $< \sqrt{d}/2$. Each C_{ij} is a set-multilinear formula over a subset $(X(w_p) : p \in S_j)$ for some $S_j \subseteq [d]$, where $S_1, \ldots, S_{t_i}$ partition $[d]$. Let $w^{i1}, \ldots, w^{it_i}$ be the corresponding decomposition of w. That is, $w^{ij} = w_{|S_j}$. Recall that for a word w^{ij} we defined in the preliminaries $w^{ij}_{S_j}$ as the sum of its entries.
  Let $j \in [t_i]$. Let a_{ij} be the number of positive indices in w^{ij}. If $2a_{ij} \leq \deg(C_{i,j})$, then

$$w^{ij}_{S_j} \leq -\frac{k}{2\sqrt{d}} \deg(C_{i,j}).$$

  Otherwise, we have

$$\begin{aligned}
\left|w^{ij}_{S_j}\right| &= \left| a_{ij} \left| k - \frac{k}{\sqrt{d}} \right| - \left(\deg(C_{i,j}) - a_{ij}\right) k \right| \\
&= \left| a_{ij} k - a_{ij} \left\lfloor \frac{k}{\sqrt{d}} \right\rfloor - \deg(C_{i,j}) k + a_{ij} k \right| \\
&> \left(2a_{ij} - \deg(C_{i,j}) - \frac{a_{ij}}{\sqrt{d}} \right) k - a_{ij} \\
&> \frac{k}{2} - \frac{k}{4} = \frac{k}{2} \cdot \frac{1}{2} \\
&> \frac{k \deg(C_{i,j})}{2\sqrt{d}}.
\end{aligned}$$

  So in both cases, $\left|w^{ij}_{S_j}\right| \geq \frac{k\deg(C_{i,j})}{2\sqrt{d}}$.
  In particular,

$$\mathrm{relrk}_w(C_i) \leq \prod_{j=1}^{t_i} 2^{-\frac{1}{2}\left|w^{ij}_{S_j}\right|} \leq 2^{-\frac{kd}{4\sqrt{d}}} \leq s_i 2^{-\frac{k\sqrt{d}}{8}}.$$

The result of the claim directly follows from the subadditivity of the measure. □

4. WHAT HAPPENS FOR LARGER CONSTANT DEPTHS

To prove Theorem 1, Theorem 3, and Corollary 4 in their generality (i.e., for any constant depths), we generalize the depth-3 approach detailed above. To do that, we just need to generalize Proposition 5 and Lemma 8 for a larger depth. We will just sketch here the ideas to obtain such a generalization. The detailed proof of the main theorem can be found in the full version of this paper.

4.1. The set-multilinearization transformation

As already noticed the set-multilinearization from homogeneous circuits transformation (Lemma 7) still works for large depths. The only point there is to generalize Lemma 6 to a larger depth. Unlike the case of depth-3, this homogenization step was not known before this work. Now, inputs of a product can have now polynomials of any degree. The idea is to consider now some weighted versions of the elementary symmetric polynomials to take care of these degrees. The main difficulty now is to check that the *Newton identities* (which play a key role in the proof of Lemma 6) still hold in this weighted setting. But finally, we can obtain the following homogeneization result.

Lemma 12. *Let Δ be an odd integer. Let s, N, d be growing parameters with $s \geq N$. If C is a circuit of size at most s and depth at most Δ computing a homogeneous polynomial $P(x_1, \ldots, x_N)$ of degree d, then, P can also be computed by a* homogeneous *circuit $\tilde{C}$ of size at most* $\mathrm{poly}(s)2^{O(\sqrt{d})}$ *and depth at most* $2\Delta - 1$.

4.2. Lower bounds for set-multilinear circuits

Then, we follow a similar path to obtain lower bounds on the size of set-multilinear circuits of any constant depth computing lopsided polynomials. In particular, we still consider the partial derivative measure. The idea now is to refine the alphabet which is used (taking values which depend on the depth Δ) so that the case where C_i is of type 2 in the proof can be tackled. Then the first case (C_i is of type 1) will correspond to an induction step.

Putting these steps in a more precise way, we can obtain the following generalization of Proposition 5 and so obtain a complete proof of Theorem 1.

Lemma 13. *Let Δ be an odd integer and let $\Gamma = (\Delta - 1)/2$. Let n, d, $\Delta \in \mathbb{N} \setminus \{0\}$ with $n \geq 2^{10d+1}$. Any set-multilinear circuit C of depth Δ computing* $\mathrm{IMM}_{n,d}$ *has size at least*

$$n^{\Omega\left(\frac{d^{1/(2^\Gamma - 1)}}{\Gamma}\right)}.$$

Acknowledgments

The authors are grateful to R. Saptharishi for spending time to discuss the proof with us. Nutan Limaye was partially funded by SERB project File no. MTR/2017/000909. Sébastien Tavenas was funded by the project BIRCA from the IDEX of Univ. Grenoble Alpes (Contract 183459). He also benefited from accommodation paid by IIT Bombay during a visit of the other two authors.

References

1. Agrawal, M., Vinay, V. Arithmetic circuits: A chasm at depth four. In *Proceedings of Foundations of Computer Science (FOCS)*, USA (2008), 67–75. DOI: 10.1109/FOCS.2008.32.
2. Beame, P., Huynh, T., Pitassi, T. Hardness amplification in proof complexity. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010*. L.J. Schulman, ed. ACM, NY, 2010, 87–96.
3. Bürgisser, P. Cook's versus valiant's hypothesis. *Theor. Comput. Sci. 235*, 1 (2000), 71–88.
4. Bürgisser, P., Clausen, M., Shokrollahi,

M.A. *Algebraic Complexity Theory*, volume 315 of *Grundlehren der Mathematischen Wissenschaften*. Springer, Berlin, Heidelberg, 1997. DOI: 10.1007/978-3-662-03338-8.
5. Chou, C.-N., Kumar, M., Solomon, N. Closure results for polynomial factorization. *Theory Comput.* 15: Paper No. 13, 34 (2019).
6. Dvir, Z., Shpilka, A., Yehudayoff, A. Hardness-randomness tradeoffs for bounded depth arithmetic circuits. *SIAM J. Comput. 39*, 4 (2009), 1279–1293.
7. Fournier, H., Limaye, N., Malod, G., Srinivasan, S. Lower bounds for depth-4 formulas computing iterated matrix multiplication. *SIAM J. Comput. 44*, 5 (2015), 1173–1201.
8. Gupta, A., Kamath, P., Kayal, N., Saptharishi, R. Approaching the chasm at depth four. *J. ACM 61*, 6 (Dec. 2014), 33:1–33:16.
9. Gupta, A., Kamath, P., Kayal, N., Saptharishi, R. Arithmetic circuits: A chasm at depth 3. *SIAM J. Comput. 45*, 3 (2016), 1064–1079.
10. Hrubeš, P., Yehudayoff, A. Homogeneous formulas and symmetric polynomials. *Comput. Complexity 20*, 3 (2011), 559–578.
11. Kabanets, V., Impagliazzo, R. Derandomizing polynomial identity tests means proving circuit lower bounds. *Comput. Complexity 13*, 1–2 (2004), 1–46.
12. Koiran, P. Arithmetic circuits: The chasm at depth four gets wider. *Theor. Comput. Sci. 448* (2012), 56–65.
13. Kumar, M., Saptharishi, R. Hardness-randomness tradeoffs for algebraic computation. *Bull. EATCS 129* (2019).
14. Kumar, M., Saraf, S. On the power of homogeneous depth 4 arithmetic circuits. *SIAM J. Comput. 46*, 1 (2017), 336–387.
15. Nisan, N., Wigderson, A. Lower bounds on arithmetic circuits via partial derivatives. *Comput. Complexity 6*, 3 (1997), 217–234.
16. Raz, R. Multi-linear formulas for permanent and determinant are of super-polynomial size. *J. ACM 56*, 2 (2009), 8:1–8:17.
17. Raz, R. Elusive functions and lower bounds for arithmetic circuits. *Theory Comput. 6*, 1 (2010), 135–177.
18. Raz, R. Tensor-rank and lower bounds for arithmetic formulas. *J. ACM 60*, 6 (2013), 40:1–40:15.
19. Saptharishi, R. A survey of lower bounds in arithmetic circuit complexity. Github survey (2015).
20. Saxena, N. Progress on polynomial identity testing. *Bull. EATCS 99* (2009), 49–79.
21. Shpilka, A., Wigderson, A. Depth-3 arithmetic circuits over fields of characteristic zero. *Comput. Complexity 10*, 1 (2001), 1–27.
22. Shpilka, A., Yehudayoff, A. Arithmetic circuits: A survey of recent results and open questions. *Found. Trends Theor. Comput. Sci. 5* (Mar 2010), 207–388.
23. Strassen, V. Vermeidung von divisionen. *J. für die reine und angewandte Mathematik 264* (1973), 184–202.
24. Tavenas, S. Improved bounds for reduction to depth 4 and depth 3. *Inf. Comput. 240* (2015), 2–11.
25. Valiant, L.G. Completeness classes in algebra. In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30–May 2, 1979, Atlanta, Georgia, USA*. M.J. Fischer, R.A. DeMillo, N.A. Lynch, W.A. Burkhard, A.V. Aho, eds. ACM, NY, 1979, 249–261.

Nutan Limaye (nuli@itu.dk), ITU Copenhagen, Denmark.

Srikanth Srinivasan (srikanth@cs.au.dk), Aarhus University, Denmark.

Sébastien Tavenas (sebastien.tavenas@univ-smb.fr), Université Savoie Mont Blanc, CNRS, LAMA, France.

DOI:10.1145/3631339

To view the accompanying paper, visit doi.acm.org/10.1145/3610801

Technical Perspective
Bridging AI with Real-Time Systems

By Giorgio Buttazzo

ARTIFICIAL INTELLIGENCE (AI) and machine learning models are making progress at an unprecedented rate and have achieved remarkable performance in several specific tasks such as image classification, object detection, automatic control, strategy games, some types of medical diagnoses, and music composition.

The exceptional performance of machine learning models in perception tasks makes them very attractive for being adopted in a large variety of autonomous systems, which must process sensory data to understand the environment and react in real time to accomplish a given task. Examples of such autonomous systems include self-driving cars, advanced robots operating in unknown environments, and interplanetary space probes. These systems must not only perceive the objects in the scene and their location with a high accuracy, but they also must predict their trajectories and plan proper actions within stringent timing constraints.

Consider, for instance, an autonomous car driving in an urban environment. Its onboard perception system is not only in charge of detecting the road, sidewalks, traffic lights, and road signs, but it is also responsible for identifying and recognizing moving objects, such as pedestrians, bicycles, and other moving vehicles, while predicting their trajectories and planning proper actions to prevent possible impacts with them. In this context, a correct prediction produced too late could cause the system to fail. This example illustrates that guaranteeing a timely response in this type of system is as crucial as producing a correct prediction.

In a complex, highly dynamic scenario like the one considered for a self-driving car, however, not all computational tasks are equally important. For example, objects closer to the vehicle should receive a higher priority with respect to those located further away. Similarly, objects moving at higher speed should be processed at higher rates with respect to objects that are standing or moving at lower speed.

One problem with the current AI frameworks and hardware accelerators for deep neural networks is that they have been developed for non-critical applications where timing is not an issue. Consequently, when multiple neural models must be executed on the same platform, each model is normally executed non-preemptively (that is, without interruption) or, in the best case, using simple scheduling heuristics that do not take time requirements or task criticality into account.

This means if a highly critical task H is activated just after a low-critical task L has started its execution, H will experience a long delay, since it can only start executing after the completion of L. This phenomenon is referred to as a *priority inversion* and has been studied extensively in the field of real-time systems. However, it represents a serious problem in current AI algorithms, preventing their use in safety-critical real-time systems, where timing and functional requirements are equally important.

> The following paper proposes new methodology for overcoming the limitations of AI frameworks to enable the use of deep neural networks in mission-critical systems.

The following paper proposes a new methodology for overcoming the limitations of current AI frameworks to enable the use of deep neural networks in mission-critical systems. The key idea is to split the perception process into multiple tasks associated with different objects and prioritize them to enable more timely response to more critical stimuli.

The system combines range data acquired from a light detection and ranging sensor (LiDAR) with images obtained from a camera. In particular, the 3D objects detected by the LiDAR (based on distances) are projected on the 2D-image plane of the camera. Then, bounding boxes are assigned a priority inversely proportional to their distance from the vehicle, so that closer objects will be attended first. In this way, depending on the overall workload, low-priority objects can be processed by a lower rate.

To overcome the limitation of non-preemptive execution, the deep neural network in charge of processing the cropped images is broken into stages, each consisting of multiple neural layers so the model inference can be preempted between stages. To add flexibility, multiple predictions with different precision are generated at the end of multiple stages to balance accuracy vs. execution time.

The overall system is then able to schedule the various perceptual tasks based on the assigned priority, while avoiding priority inversion during neural inference and enabling a more predictable execution of AI algorithms in mission-critical systems.

Giorgio Buttazzo is a professor of computer engineering at the Sant'Anna School of Advanced Studies in Pisa, Italy.

DOI:10.1145/3610801

To view the accompanying Technical Perspective, visit doi.acm.org/10.1145/3631339

Taming Algorithmic Priority Inversion in Mission-Critical Perception Pipelines

By Shengzhong Liu, Shuochao Yao, Xinzhe Fu, Rohan Tabish, Simon Yu, Ayoosh Bansal, Heechul Yun, Lui Sha, and Tarek Abdelzaher

Abstract

The paper discusses *algorithmic priority inversion* in mission-critical machine inference pipelines used in modern neural-network-based perception subsystems and describes a solution to mitigate its effect. In general, *priority inversion* occurs in computing systems when computations that are "less important" are performed together with or ahead of those that are "more important." Significant priority inversion occurs in existing machine inference pipelines when they do not differentiate between critical and less critical data. We describe a framework to resolve this problem and demonstrate that it improves a perception system's ability to react to critical inputs, while at the same time reducing platform cost.

1. INTRODUCTION

Algorithmic priority inversion plagues modern *mission-critical* machine inference pipelines such as those implementing perception modules in autonomous drones and self-driving cars. We describe an initial solution for removing such priority inversion from neural-network-based perception systems. This research was originally published in RTSS 2020.[17] While it is evaluated in the context of autonomous driving only, the design principles described below are expected to remain applicable in other contexts.

The application of artificial intelligence (AI) has revolutionized cyber-physical systems but has posed novel challenges in aligning computational resource consumption with mission-specific priority. Perception is one of the key components that enable system autonomy. It is also a major efficiency bottleneck that accounts for a considerable fraction of resource consumption.[3,12] In general, priority inversion occurs in computing systems when computations that are less critical (or that have longer deadlines) are performed together with or ahead of those that are more critical (or that have shorter deadlines). Current neural-network-based machine intelligence software suffers from a significant form of priority inversion on the path from perception to decision-making, because it processes input data sequentially in arrival order as opposed to processing important parts of a scene first. By resolving this problem, we significantly improve the system's responsiveness to critical inputs at a lower platform cost. The work applies to intelligent systems that perceive their environment in real-time (using neural networks), such as self-driving vehicles,[1] autonomous delivery drones,[5] military defense systems,[2] and socially-assistive robotics.[8]

To understand the present gap, observe that current deep perception networks perform many layers of manipulation of large multidimensional matrices (called *tensors*). The underlying neural network libraries (e.g., *TensorFlow*) are reminiscent of what used to be called the *cyclic executive*[4] in early operating system literature. Cyclic executives, in contrast to priority-based scheduling,[11] processed all pieces of incoming data at the same *priority* and *fidelity* (e.g., as nested loops). Given incoming data frames (e.g., multicolor images or 3D LiDAR point clouds), modern neural network algorithms process all data rows and columns at the same priority and fidelity. Importance cues drive attention weights in AI computations, but not actual computational resource assignments.

This flat processing is in sharp contrast to the way *humans* process information. Human cognitive perception systems are good at partitioning the perceived scene into semantically meaningful partial regions in real-time, before allocating different degrees of attention (i.e., processing fidelity) and prioritizing the processing of important parts, to better utilize the limited cognitive resources. Given a complex scene, such as a freeway with multiple nearby vehicles, human drivers are good at understanding what to focus on to plan a valid path forward. In fact, human cognitive capacity is not sufficient to simultaneously absorb everything in their field of view. For example, if faced with an iMax screen partitioned into a dozen subdivisions, each playing an independent movie, humans would be fundamentally incapable of giving all such simultaneously playing movies sufficient attention. This suggests that GPUs that can, in fact, keep up with processing all pixels of the input scene are fundamentally and needlessly over-provisioned. They could be substantially smaller

The original version of the article, "On Removing Priority Inversion from Mission-Critical Machine Inference Pipelines" was published in the *Proceedings of the IEEE 2020 Real-Time Systems Symposium*.

if endowed with a human-like capacity to focus on part of the scene only. The lack of prioritized allocation of processing resources to different parts of an input data stream (e.g., from a camera) is an instance of *algorithmic priority inversion*. As exemplified above, it results in significant resource waste, processing less important stimuli together with more important ones. To avoid wasting resources, the architecture described in this paper allows machine perception pipelines to partition the scene into regions of different criticality, prioritize the processing of important parts ahead of others, and provide higher processing fidelity on critical regions.

2. SYSTEM ARCHITECTURE

Consider a simple pipeline composed of a camera that observes its physical environment, a neural network that processes the sampled frames, and a control unit that must react in real-time. Figure 1 contrasts the traditional design of such a machine inference pipeline to the proposed architecture. In the traditional design, the captured input data frames are processed sequentially by the neural network without preemption in execution.

Unfortunately, the multi-dimensional data frames captured by modern sensors (e.g., colored camera images and 3D LiDAR point clouds) carry information of different degrees of criticality in every frame.[a] Data of different criticality may require a different processing latency. For example, processing parts of the image that represent faraway objects does not need to happen every frame, whereas processing nearby objects, such as a vehicle in front, needs to be done immediately because of their impact on immediate path planning. To accommodate these differences in input data criticality, our machine perception pipeline breaks the input frame processing into four steps:

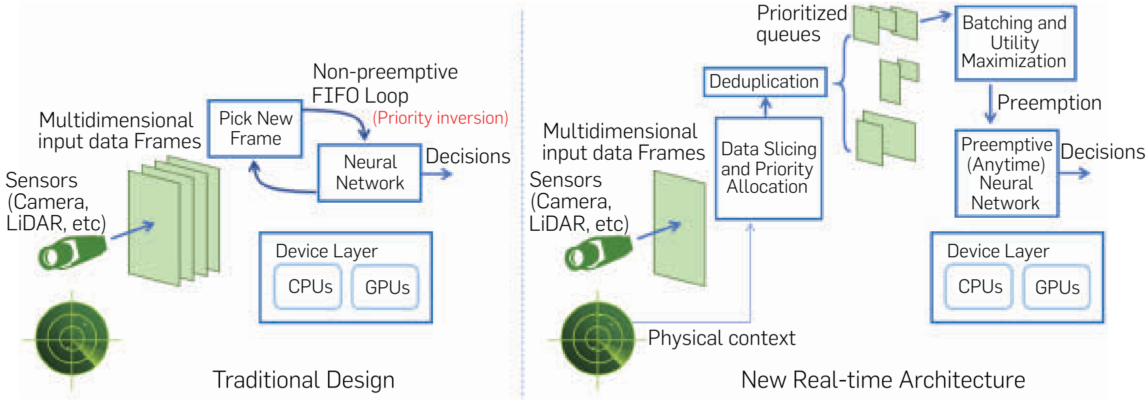
- Data slicing and priority allocation: This module breaks up newly arriving frames into smaller regions of different degrees of criticality based on simple heuristics (i.e., distance-based criticality).
- Deduplication: This module drops redundant regions (i.e., ones that refer to the same physical objects) across successive arriving frames.
- "Anytime" neural network: This neural network implements an imprecise computation model that allows execution to be preempted while yielding partial utility from the partially completed computation. The approach allows newly arriving critical data to preempt the processing of less critical data from older frames.
- Batching and utility maximization: This module sits between the data slicing and deduplication modules on one end and the neural network on the other. With data regions broken by priority, it decides which regions to pass to the neural network for processing. Since multiple regions may be queued for processing, it also decides how best to benefit from batching (that improves processing efficiency).

We refer to the subsystem shown in Figure 1 as the *observer*. The goal is to allow the observer to respond to more urgent stimuli ahead of less urgent ones. To make the observer concrete, we consider a video processing pipeline, where the input video frames get broken into regions of different criticality according to the distance information obtained from a ranging sensor (i.e., LiDAR). Different deadline-driven priorities are then assigned to the processing of these regions. We adopt an imprecise computation model for neural networks[21] to achieve a hierarchy of different processing fidelities. We further introduce a utility-optimizing scheduling algorithm for the resulting real-time workload to meet deadlines while maximizing a notion of global utility (to the mission). We implement the architecture on an NVIDIA Jetson Xavier platform and do a performance evaluation on the platform using real video traces collected from autonomous vehicles. The results show that the new algorithms significantly improve the average quality of machine inference, while nearly eliminating deadline misses, compared to a set of state-of-the-art baselines executed on the same hardware under the same frame rate.

For completeness, below we first describe all components of the observer, respectively. We then detail the batching and utility maximization algorithm used.

a By different degrees of *criticality*, we are referring to different levels of importance within the *mission-critical* sub-system. For example, faraway objects are less relevant to path planning than nearby objects.

Figure 1. Real-time machine inference pipeline architecture.



2.1. Data slicing and priority allocation

This module breaks up input data frames into regions that require different degrees of attention. Objects with a smaller *time-to-collision*[18] should receive attention more urgently and be processed at a higher fidelity. We further assume that the observer is equipped with a *ranging* sensor. For example, in autonomous driving systems, a LiDAR sensor measures distances between the vehicle and other objects. LiDAR point cloud-based object localization techniques have been proposed[6] that provide a fast (i.e., over 200Hz) and accurate ranging and object localization capability. The computed object locations can then be projected onto the image obtained from the camera, allowing the extraction of regions (subareas of the image) that represent these localized objects, sorted by distance from the observer. For simplicity, we restrict those subareas to rectangular regions or *bounding boxes*. We define the priority (of bounding boxes) by time-to-collision, given the trajectory of the observer and the location of the object. Computing the time-to-collision is a well-studied topic and is not our contribution.[18]

2.2. Deduplication

The deduplication module eliminates redundant bounding boxes. Since the same objects generally persist across many frames, the same bounding boxes will be identified in multiple frames. The set of bounding boxes pertaining to the same object in different frames is called a *tubelet*. Since the best information is usually the most recent, only the most recent bounding box in a tubelet needs to be acted on. The deduplication module identifies boxes with large overlaps as redundant and stores the most recent box only. For efficiency reasons described later, we quantize the used bounding box sizes. The deduplication module uses the same box size for the same object throughout the entire tubelet. Note that, in a traditional neural network processing pipeline, each frame is processed in its entirety before the next one arrives. Thus, no deduplication module is used. The option to add this time-saving module to our architecture arises because our pipeline can postpone the processing of some objects until a later time. By that time, updated images of the same object may arrive. This enables savings by looking at the latest image only when the neural network eventually gets around to processing the object.

2.3. The anytime neural network

A perfect *anytime* algorithm is one that can be terminated at any point, yielding utility that monotonically increases with the amount of processing performed. We approximate the optimal model with an imprecise computation model,[14–16] where the processing consists of two parts: a *mandatory part* and multiple *optional parts*. The optional parts, or a portion thereof, can be skipped to conserve resources. When at least one optional part is skipped, the task is said to produce an *imprecise* result. Deep neural networks (e.g., image recognition models[10]) are a concatenation of a large number of layers that can be divided into several stages, as we show in Figure 2. Ordinarily, an output layer is used at the end to convert features computed by earlier layers into the output value (e.g., an object classification). Prior work has shown, however, that other output layers can be forked off of intermediate stages producing meaningful albeit imprecise outputs based on features computed up to that point.[20] Figure 3 shows the accuracy of ResNet-based classification applied to the ImageNet[7] dataset at the intermediate stages of neural network processing. The quality of outputs increases when the network executes more optional parts. We set the utility proportionally to *predictive confidence in result*; a low confidence output is less useful than a high confidence output. The proportionality factor itself can be set depending on task criticality, such that uncertainty in the output of more critical tasks is penalized more.

2.4. Batching and utility maximization

This module decides the schedule of processing of all regions identified by the data slicing and prioritization module and that passes de-duplication. The data slicing module computes *bounding boxes* for objects detected, which constitute regions that require attention, each assigned a degree of criticality. The deduplication module groups boxes related to the same object into a tubelet. Only the latest box in the tubelet is kept. Each physical object gives rise to a separate neural network task to be scheduled. The input of that task is the bounding box for the corresponding object (cropped from the full scene).

3. THE SCHEDULING PROBLEM

In this section, we describe our task execution model, formulate the studied scheduling problem, and derive a near-optimal solution.

3.1. The execution model

As alluded to earlier, the scheduled tasks in our system constitute the execution of multi-layer deep neural networks (e.g., ResNet,[10] as shown in Figure 2), each processing a different

Figure 2. ResNet[10] architecture with multiple exits. On the left, we show the design of the basic bottleneck block of ResNet. *c* is the feature dimension. The classifier has a pooling layer and a fully connected layer.

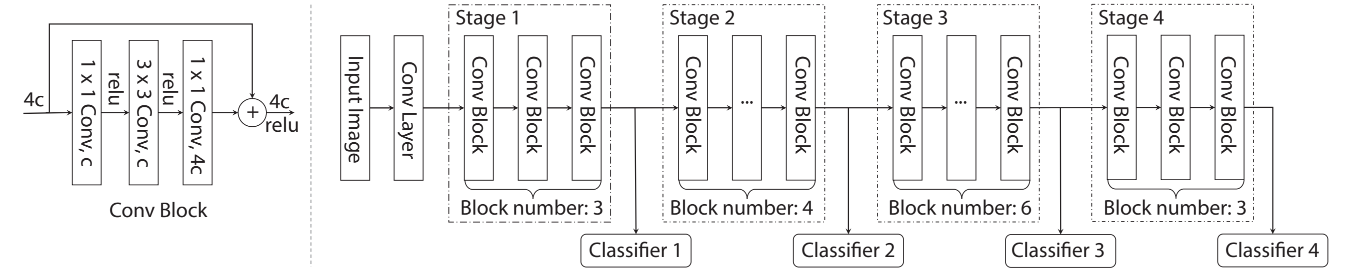
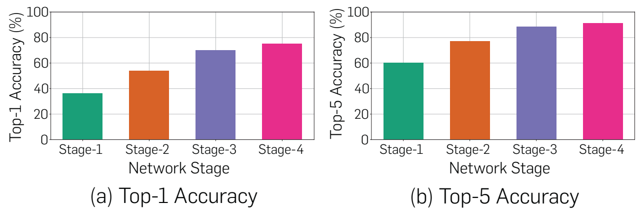


Figure 3. ResNet stage accuracy change on ImageNet[7] dataset.



input data region (i.e., a bounding box). As shown in Figure 2, tasks are broken into stages, where each stage includes multiple neural network layers. The unit of scheduling is a single stage, whose execution is non-preemptive, but tasks can be preempted on stage boundaries. A task arrives when a new object is detected by the ranging sensor (e.g., LiDAR) giving rise to a corresponding new bounding box in the camera scene. Let the arrival time of task τ_i be denoted by a_i. A deadline $d_i > a_i$, is assigned by the data slicing and priority assignment module denoting the time by which the task must be processed (e.g., the corresponding object classified). The data slicing and priority assignment module are invoked at frame arrival time. Therefore, both a_i and d_i are a multiple of frame inter-arrival time, H. No task can be executed after its deadline. Future object sizes, arrival times, and deadlines are unknown, which makes the scheduling problem an *online decision problem*. A combination of two aspects makes this real-time scheduling problem interesting: *batching* and *imprecise computations*. We describe these aspects below.

Batching. Stages of the neural network, in our architecture, are executed on a low-end embedded GPU. While such GPUs feature parallel execution, most require that the same kernel be executed on all GPU cores. This means that we can process different images concurrently on the GPU as long as we run the *same kernel* on all GPU cores. We call such concurrent execution, *batching*. Running the same kernel on all GPU cores means that we can only batch image processing tasks if both of the following apply: (i) they are executing *the same neural network stage,* and (ii) they *run on the same size inputs*. The latter condition is because the processing of different bounding box sizes requires instantiating different GPU kernels. Batching is advantageous because it allows us to better utilize the parallel processing capacity of GPU. To increase batching opportunities, we limit the size of possible bounding boxes to a finite set of options. For a given bounding box size k, at most $B^{(k)}$ tasks (processing inputs) can be batched together before overloading the GPU capacity. We call it the *batching limit* for the corresponding input size.

Imprecise computations. Let the number of neural network stages for task τ_i be L_i (different input sizes may have different numbers of stages). We call the first stage *mandatory* and call the remaining stages *optional*. Following a recently developed imprecise computation model for deep neural networks (DNN),[21] tasks are written such that they can return an object classification result once the mandatory stage is executed. This result then improves with the execution of each optional stage. Earlier work presented an approach to estimate the expected confidence in the correctness of the results of future stages, ahead of executing these stages.[22] This estimation offers a basis for assessing the utility of future task stage execution. We denote the utility of task τ_i after executing $j \le L_i$ stages by $R_{i,j}$, where $R_{i,j}$ is set proportionately to the predicted confidence in correctness at the conclusion of stage j. Note that, the expected utility can be different among tasks (depending in part on input size), but it is computable, non-decreasing, and concave with respect to the network stage.[22]

We denote by $\mathcal{T}(t)$ the set of *current tasks* at time t. A task, τ_i, is called *current* at time t, if $a_i \le t < d_i$, and the task has not yet completed its last stage, L_i. For task τ_i of input size, k, the execution time of the j-th stage is denoted by $e^{(k)}_{j,b}$, where b is the number of batched tasks during the stage execution.

3.2. Problem formulation

We next formulate a new scheduling problem, called *BAtched Scheduling with Imprecise Computations (BASIC)*. The problem is simply to decide on the number of stages $l_i \le L_i$ to execute for each task τ_i and to schedule the batched execution of those task stages on the GPU such that the total utility, $\sum_i R_{i,l_i}$, of executed tasks is maximized, and batching constraints are met (i.e., all used GPU cores execute the same kernel at any given time, and that the batching limit is not exceeded). In summary:

The BASIC problem. *With online task arrivals, the objective of the BASIC problem is to derive a schedule x to maximize the aggregate system utility. The schedule decides three outputs: task stage execution order on the GPU, number of stages to execute for each task, and task batching decisions. For each scheduling period t, we use* $x_t(i,j) \in \{0,1\}$ *to denote whether the j-th stage of task* τ_i *is executed. Besides, we use P to denote a batch of tasks, where* $|P|$ *denotes the number of tasks being batched. The mathematical formulation of the optimization problem is:*

$$BASIC: \ \max \sum_{x_t}\sum_{i} x_t(i,j)\left(R_{i,j}-R_{i,j-1}\right)$$

$$\text{s.t.} \quad x_t(i,j)\in\{0,1\}, \ \sum_{t=1}^{T} x_t(i,j)\le 1, \ \ \forall i,j \qquad (1)$$

$$x_t(i,j)=0, \ \ \forall t \notin [a_i, d_i), \ \ \forall i,j \qquad (2)$$

$$\sum_{t'=1}^{t-1} x_{t'}(i,j-1)-x_t(i,j)\ge 0, \quad \forall i, j>1, t>1 \qquad (3)$$

$$s_i = s_{i'} = k, \ \ l_i = l_{i'}, \ \ |P|\le b_k, \quad \forall i\in P, \ i'\in P, \ \exists k\in\mathcal{S} \qquad (4)$$

The following constraints should be satisfied: (1) Each neural network stage can only be executed once; (2) No task can be executed after its deadline; (3) The execution of different stages of the same task must satisfy their precedence constraints; and (4) Only tasks with the same (image size, network stage) can be batched, and the number of batched tasks can not exceed the batching constraint of their image size.

Only one batch (kernel) can be executed on the GPU at any time. However, multiple batches can be executed sequentially in one scheduling period, as long as the sum of their execution times does not exceed the period length, H.

3.3. An online scheduling framework

We derive an optimal dynamic programming-based solution for the BASIC scheduling problem and express its competitive ratio relative to a clairvoyant scheduler (that has full knowledge of all future task arrivals). We then derive a more efficient greedy algorithm that approximates the dynamic programming schedule. We define the clairvoyant scheduling problem as follows:

DEFINITION 1 (CLAIRVOYANT SCHEDULING PROBLEM). *Given information about all future tasks, the clairvoyant scheduling problem seeks to maximize the aggregate utility obtained from (stages of) tasks that are completed before their deadlines. The maximum aggregate utility is OPT.*

With no future information, an online scheduling algorithm that achieves a competitive ratio of c (i.e., a utility $\geq \frac{1}{c} \cdot OPT$) is called c-competitive. A lower bound on the competitive ratio for online scheduling algorithms was shown to be 1.618.[9]

Our scheduler is invoked upon frame arrivals, which is once every H unit of time. We thus call H the *scheduling period*. We assume that all task stage execution times are multiples of some basic time unit δ, thereby allowing us to express H by an integer value. We further call the problem of scheduling current tasks within the period between successive frame arrivals, the *local scheduling problem*:

DEFINITION 2 (LOCAL BASIC PROBLEM). *Given the set of current tasks, $\mathcal{T}(t)$, within the scheduling period, t, the local BASIC problem seeks to maximize the total utility gained within this scheduling period only.*

We proceed to show that an online scheduling algorithm that optimally solves the local scheduling problem within each period will have a good competitive ratio. Let L be the maximum number of stages in any task, and let B be the maximum batching size:

THEOREM 1. *If during each scheduling period, the local BASIC problem for that period is solved optimally, then the resulting online scheduling algorithm is* $\min\{2 + L, 2B + 1\}$*-competitive with respect to a clairvoyant algorithm.*

When no imprecise computation is considered, the competitive ratio is further reduced to:

COROLLARY 1. *If each task is only one stage long, and if the online scheduling algorithm solved the local BASIC problem in each scheduling period optimally, then the online scheduling algorithm is 3-competitive with respect to a clairvoyant algorithm.*

3.4. Local scheduling algorithms

In this section, we propose two algorithms to solve the local BASIC problem. The first is a dynamic programming-based algorithm that optimally solves it but may have a higher computational overhead. The second is a greedy algorithm that is computationally efficient but may not optimally solve the problem.

Local dynamic programming scheduling. Since we only consider batching together on the GPU tasks that execute the same kernel (i.e., same stage on the same size input), we need to partition the scheduling interval, H, into sub-intervals where the above constraint is met. The challenge is to find optimal partitioning. This question is broken into three steps:

- Step 1: Given an amount of time, $T_{j,k} \leq H$, what is the maximum utility attainable by scheduling the same stage, j, of tasks that process an input of size k? The answer here simply depends on the maximum number of tasks that we can batch during $T_{j,k}$ without violating the batching limit. If the time allows for more than one batch, dynamic programming is used to optimally size the batches. Let the maximum attainable utility thus found be denoted by $U^*_{j,k}$.
- Step 2: Given an amount of time, $T_k \leq H$, what is the maximum utility attainable by scheduling (any number of stages of) tasks that process an input of size k? Let us call this maximum utility U^*_k. Dynamic programming is used to find the best way to break interval T_k into non-overlapping intervals $T_{j,k}$, for which the total sum of utilities, $U^*_{j,k}$, is maximum.
- Step 3: Given the scheduling interval, H, what is the maximum utility attainable by scheduling tasks of different input sizes? Let us call this maximum utility U^*. Dynamic programming is used to find the best way to break interval H into non-overlapping intervals T_k, for which the total sum of utilities, U^*_k, is maximum.

The resulting utility, U^*, as well as the corresponding breakdown of the scheduling interval constitute the optimal solution. In essence, the solution breaks down the overall utility maximization problem into a utility maximization problem over time sub-intervals, where tasks process only a given input size. These sub-intervals are in turn broken into sub-intervals that process the same stage (and input size). The intuition is that the sub-intervals in question do not overlap. We pose an *order preserving* assumption on task marginal utilities with the same image size.

ASSUMPTION 1 (ORDER PRESERVING ASSUMPTION). *For two tasks τ_{i_1} and τ_{i_2} with the same size, if for one neural network stage j, we have $R_{i_1,j} - R_{i_1,j-1} \geq R_{i_2,j} - R_{i_2,j-1}$, then it also holds $R_{i_1,j+1} - R_{i_1,j} \geq R_{i_2,j+1} - R_{i_2,j}$.*

Thus, the choice of the best subset of tasks to execute remains the same regardless of which stage is considered. Below, we describe the algorithm in more detail.

Step 1: For each object size k and stage j, we can use a dynamic programming algorithm to decide the maximum number of tasks M that can execute stage j in time $0 < T_{j,k} \leq H$. Observe that this computation can be done offline. The details are shown in Algorithm 1. With the optimal number, M, computed for each, $T_{j,k}$, $U^*_{j,k}$ is simply the sum of utilities of the M highest-utility tasks that are ready to execute stage j on an input of size k.

Step 2: We solve this problem by two-dimensional dynamic programming, considering the considered network stages and the time, respectively. The recursive (induction) step takes the output of Step 1 as input to calculate the optimal utility from assigning some fraction of T_k to the first $j-1$ stage and the remainder to stage j, and computes the best possible sum of the two, for each T_k. Once all stages are considered, the result is the optimal utility, U_k^*, from running tasks of input size k for a period T_k. The details are explained in Algorithm 2.

Algorithm 1: Batching

Input: Image size index k, stage j, execution time e_b, batching constraint B, period H.
Output: Maximum achievable tasks $M(h)$, and optimal batch sequence $P(h)$, $\forall h \le H$.

$M(h) = 0$, $P(h) = \emptyset$, $\forall 0 \le h \le H$;
for $b = 1, \dots, B$ **do**
  if $b > M(e_b)$ **then**
    $M(e_b) := b$, $P(e_b) := \{(k, j, b)\}$;
  end
end
for $h = 2, \dots, H$ **do**
  $h' = \arg\max_{0 \le h' \le h} M(h') + M(h - h')$;
  $M(h) := M(h') + M(h - h')$;
  $P(h) := P(h') \cup P(h - h')$;
end
return M, P.

Algorithm 2: Stage Assignment

Input: Maximum tasks M, optimal batch sequence P, available task set $\mathcal{T}_j$ for each stage j, stage count L, period H.
Output: Maximum achievable utilities U_{OPT}, and optimal batch sequence P_{OPT}, $\forall h \le H$.

$U_{OPT}(j, h) = 0$, $P_{OPT}(j, h) = \emptyset$, $\forall j, h$;
Transitted object buffer $\mathcal{T}(j, h) = \emptyset$, $\forall j, h$;
for $j = 1, \dots, L$ **do**
  for $h = 1, \dots, H$ **do**
    if $j = 1$ **then**
      $n := \min(M(j, h), |\mathcal{T}_j|)$;
      $\mathcal{T}(j, h) := n$ tasks with max utility in $\mathcal{T}_j$;
      $U_{OPT}(j, h) :=$ total utility of $\mathcal{T}(j, h)$;
      $P_{OPT}(j, h) := P(j, h)$;
    end
    else
      $h' :=$ $\arg\max_{h' \le h} U_{OPT}(j-1, h') + \tilde{U}(j, h - h')$, where $\tilde{U}(j, h - h') :=$ max utility achievable with $\mathcal{T}_j \cup \mathcal{T}(j-1, h')$ in time $h - h'$;
      $\mathcal{T}(j, h) :=$ executed tasks in $\tilde{U}(j, h - h')$;
      $U_{OPT}(j, h) := U_{OPT}(j-1, h') + \tilde{U}(j, h - h')$;
      $P_{OPT}(j, h) := P_{OPT}(j-1, h') \cup P(j, h)$;
    end
  end
end
return $U_{OPT}(L, h), P_{OPT}(L, h), \forall h$.

Algorithm 3: Local DP Scheduling Algorithm

Input: Available task set $\mathcal{T}^{(k)}(t)$ for each size, maximum tasks M, optimal batch sequence P, period H.
Output: Local task schedule x_t

for $k = 1, \dots, K$ **do**
  $U_{OPT}^{(k)}$, $P_{(OPT)}^{(k)}$:=**Algorithm 2**($M, P, \mathcal{T}^{(k)}(t), H$).
end
$U_{OPT}(k, h) := U_{OPT}^{(1)}(h), \forall k, h$;
$P_{OPT}(k, h) := P_{(OPT)}^{(1)}(h), \forall k, h$;
for $k = 2, \dots, K$ **do**
  for $h = 1, \dots, H$ **do**
    $h' :=$ $\arg\max_{0 \le h' \le h} U_{OPT}(k-1, h') + U_{OPT}^{(k)}(h - h')$;
    $U_{OPT}(k, h) := U_{OPT}(k-1, h') + U_{OPT}^{(k)}(h - h')$;
    $P_{OPT}(k, h) := P_{OPT}(k-1, h') \cup P_{OPT}^{(k)}(h - h')$;
  end
end
return *The schedule* x_t *according to* $P_{OPT}(K, T)$.

Algorithm 4: Local Greedy Scheduling Algorithm

Input: Available task set $\mathcal{T}(t)$, the batch limit $B^{(k)}$ for each image index k.
Output: Local task schedule x_t

while *until the end of the period* **do**
  for $k = 1, \dots, K$ **do**
    $\mathcal{T}_k(t) :=$ set of available tasks of size k.
    if $|\mathcal{T}_k(t)| \le B^{(k)}$ **then**
      $U_k(t) :=$ total utility of tasks in $\mathcal{T}_k(t)$.
      $\tilde{\mathcal{T}}_k(t) := \mathcal{T}_k(t)$
    end
    else
      $\tilde{\mathcal{T}}_k(t) := B^{(k)}$ tasks with the maximum utility in $\mathcal{T}_k(t)$, $U_k(t) :=$ total utility of tasks in $\tilde{\mathcal{T}}_k(t)$.
    end
  end
  Execute the tasks in $\tilde{\mathcal{T}}_k(t)$ with the maximum $U_k(t)$.
end
return x_t

Step 3: Similar to Step 2, we perform a standard dynamic programming procedure to decide the optimal time partitioning among tasks processing different input sizes. The details of this procedure, along with the integrated local dynamic programming scheduling algorithm are presented in Algorithm 3.

The optimality of Algorithm 3 follows from the optimality of dynamic programming. Hence, the overall competitive ratio is 3 for single-stage task scheduling and $\min\{L+2, 2B+1\}$ for multi-stage task scheduling, according to Corollary 1 and Theorem 1, respectively. However, this algorithm may have a high computational overhead since Algorithms 2 and 3 which need to be executed each scheduling period, are $O(KLH^3)$. Next, we present a simpler local greedy algorithm, which has better time efficiency.

Local Greedy scheduling. The greedy online scheduling algorithm solves the local BASIC scheduling problem following a simple greedy selection rule: Execute the (eligible) batch with the maximum utility next. The pseudo-code of the greedy scheduling algorithm is shown in Algorithm 4. The greedy scheduling algorithm is simple to implement and has a very low computational overhead. We show that it achieves a comparable performance to the optimal algorithm in practice.

4. EVALUATION

In this section, we verify the effectiveness and efficiency of our proposed scheduling framework by comparing it with several state-of-the-art baselines on a large-scale self-driving dataset, Waymo Open Dataset.

4.1. Experimental setup

Hardware platform. All experiments are conducted on an NVIDIA Jetson AGX Xavier SoC, which is specifically designed for automotive platforms. It's equipped with an 8-core Carmel Arm v8.2 64-bit CPU, a 512-core Volta GPU, and 32GB memory. Its mode is set as MAXN with maximum CPU/GPU/memory frequency budget, and all CPU cores are online.

Dataset. Our experiment is performed on the Waymo Open Dataset,[19] which is a large-scale autonomous driving dataset collected by Waymo self-driving cars in diverse geographies and conditions. It includes driving video segments of the 20s each, collected by LiDARs and cameras at 10Hz. Only the front camera data is used in our experiment.

Neural network training. We use ResNet proposed by He *et al*.[10] for object classification. The network is trained on a general-purpose object detection dataset, COCO.[13] It contains 80 object classes that cover Waymo classes.

Scheduling load and evaluation metrics. We extract the distance between objects and the autonomous vehicle (AV) from the projected LiDAR point cloud. The deadlines of object classification tasks are set as the time to collision (TTC) with the AV. To simulate different loads for the scheduling algorithms, we manually change the sampling period (i.e., frame rate) from 40ms to 160ms. We consider a task to miss its deadline if the scheduler fails to run the mandatory part of the task by the deadline. In the following evaluation, we present both the *normalized accuracy* and *deadline miss rate* for different algorithms. The normalized accuracy is defined as the ratio between achieved accuracy and the maximum accuracy when all neural network stages are finished for every object.

4.2. Compared scheduling algorithms

The following scheduling algorithms are compared.

- OnlineDP: the online scheduling algorithm we proposed in Section 3. The local scheduling is conducted by the hierarchical dynamic programming algorithm.
- Greedy: the online scheduling algorithm we proposed, with the local scheduling conducted by the greedy batching algorithm.
- Greedy-NoBatch: It always executes the object with maximal marginal utility without batching.
- EDF: It always chooses the task stage with the earliest deadline (without considering task utility).
- Non-Preemptive EDF (NP-EDF): This algorithm does not allow preemption. It is included to understand the impact of allowing preemption on stage boundaries compared to not allowing it.
- FIFO: It runs the task with the earliest arrival time first. All stages are performed as long as the deadline is not violated.
- RR: Round-robin scheduling algorithm. Runs one stage of each task in a round-robin fashion.

4.3. Slicing and batching

We compare the inference time for *full frames* and *batched partial frames with/out deduplication*. In full-frame processing, we directly run the neural network on image-captured full images, whose size is 1920 × 1280. In *batched partial frames*, we do the slicing into bounding boxes within one frame first, then perform the deduplication (if applicable), and finally, batch execution of objects with the same size. Each frame is evaluated independently. No imprecise computation is considered. Our results show that the average latency for full frames is 350ms, while the average latency for (the sum of) batched partial frames is 105ms without deduplication, and 83ms with deduplication. Besides, the cumulative distributions of frame latencies for the three methods are shown in Figure 4. Data slicing, batching, and deduplication steps, although induce extra processing delays, can effectively reduce the end-to-end latency. However, neither approach is fast enough compared to 100ms sampling period, so that the imprecise computation model and prioritization are needed.

4.4. Scheduling policy comparisons

Next, we evaluate the scheduling algorithms in terms of achieved classification accuracy and deadline miss rate. The scheduling results are presented in Figure 5. The two proposed algorithms, OnlineDP and Greedy, clearly outperform all the

Figure 4. Cumulative distribution comparison of end-to-end latency. The execution time for frame slicing, deduplication (if applicable), batching, and neural network inference are all counted.

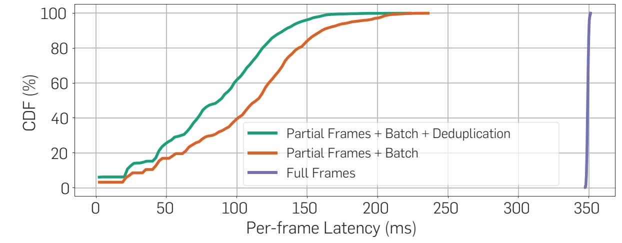
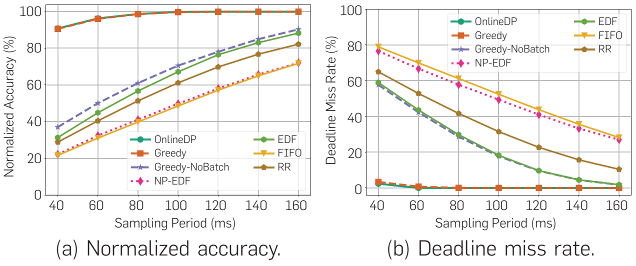


Figure 5. Accuracy and deadline miss rate comparisons on all objects.



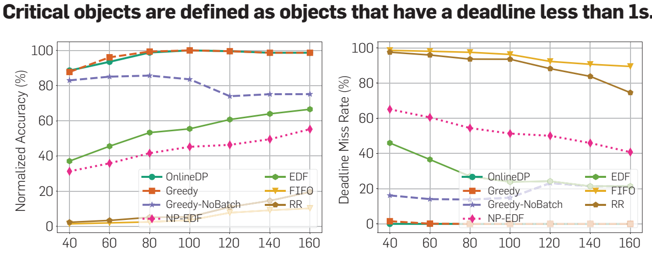
baselines with a large margin in all metrics. The improvement comes for two reasons: First, the integration of the imprecise computation model into neural networks makes the scheduler more flexible. It makes the neural network partially preemptive at the stage level, and gives the scheduler an extra degree of freedom (namely, deciding how much of each task to execute). Second, the involvement of batching simultaneously improves the model performance and alleviates deadline misses. The batching mechanism enables the GPU to be utilized at its highest parallel capacity. The deadline miss rates of both OnlineDP and Greedy are pretty close to 0 under any task load. We find Greedy shows similar performance as OnlineDP, though they possess different theoretical results. One practical reason is that the utility prediction function can not perfectly predict the utility for all future stages, where the OnlineDP scheduling can be negatively impacted.

To evaluate scheduling performance in driving scenarios involving the aforementioned important subcases, we compare the metrics of different algorithms for the subset of "critical objects." Critical objects are defined as objects whose time-to-collision (and hence processing deadline) fall within 1s from when they first appear in the scene. Results are shown in Figure 6. We notice that the accuracy and deadline miss rates of FIFO and RR are much worse in this case (because severe priority inversion occurs in these two algorithms). The deadline-driven algorithms (NP-EDF and EDF) can effectively resolve this issue because objects with earlier deadlines are always executed first. However, their general performance is limited for a lack of utility optimization. The utility-based scheduling algorithms (Greedy, Greedy-NoBatch, and OnlineDP) are also effective in removing priority inversion, while at the same time achieving better confidence in results. These algorithms multiply a weight factor $\alpha > 1$ to increase the utility of handling critical objects so that they are preferred by the algorithm over non-critical ones.

5. CONCLUSION

We presented a novel perception pipeline architecture and scheduling algorithm that resolve algorithmic priority inversion in mission-critical machine inference pipelines, prevalent in conventional FIFO-based AI workflows. To mitigate the impact of priority inversion, the proposed online scheduling architecture rests on two key ideas: (1) Prioritize parts of the incoming sensor data over others to enable a more timely response to more critical stimuli, and (2) Explore the maximum parallel capacity of the GPU by a novel task batching algorithm that improves both response speed and quality. An extensive evaluation, performed on a real-world driving dataset, validates the effectiveness of our framework.

Figure 6. Accuracy and deadline miss rate comparisons on critical objects. Critical objects are defined as objects that have a deadline less than 1s.



(a) Normalized accuracy of critical objects.

(b) Deadline miss rate of critical objects.

Acknowledgments

Research reported in this paper was sponsored in part by the Army Research Laboratory under Cooperative Agreement W911NF-17-20196, NSF CNS 18-15891, NSF CNS 18-15959, NSF CNS 19-32529, NSF CNS 20-38817, Navy N00014-17-1-2783, and the Boeing Company.

References

1. Driverless guru. https://www.driverlessguru.com/self-driving-cars-facts-and-figures. 2020.
2. Abdelzaher, T., Ayanian, N., Basar, T., Diggavi, S., Diesner, J., Ganesan, D., et al. Toward an internet of battlefield things: A resilience perspective. *Comput. 51*, 11 (2018), 24–36.
3. Alcon, M., Tabani, H., Kosmidis, L., Mezzetti, E., Abella, J., Cazorla, F.J. Timing of autonomous driving software: Problem analysis and prospects for future solutions. In *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)* (2020), IEEE, NY, 267–280.
4. Baker, T.P., Shaw, A. The cyclic executive model and ada. *Real-Time Syst. 1*, 1 (1989), 7–25.
5. Bamburry, D. Drones: Designed for product delivery. *Des. Manage. Rev. 26*, 1 (2015), 40–48.
6. Bogoslavskyi, I., Stachniss, C. Fast range image-based segmentation of sparse 3D laser scans for online operation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2016), IEEE, NY, 163–169.
7. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), IEEE, NY, 248–255.
8. Feil-Seifer, D., Matarić, M.J. Socially assistive robotics. *IEEE Rob. Autom. Mag. 18*, 1 (2011), 24–31.
9. Hajek, B. On the competitiveness of on-line scheduling of unit-length packets with hard deadlines in slotted time. In *Proceedings of the 2001 Conference on Information Sciences and Systems* (2001).
10. He, K., Zhang, X., Ren, S., Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), 770–778.
11. Lehoczky, J., Sha, L., Ding, Y. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. *RTSS 89* (1989), 166–171.
12. Lin, S.-C., Zhang, Y., Hsu, C.-H., Skach, M., Haque, M.E., Tang, L., et al. The architectural implications of autonomous driving: Constraints and acceleration. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems* (2018), 751–766.
13. Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., et al. Microsoft coco: Common objects in context. In *European Conference on Computer Vision* (2014), Springer, 740–755.
14. Liu, J.W., Lin, K.-J., Natarajan, S. Scheduling real-time, periodic jobs using imprecise results. 1987.
15. Liu, J.W., Shih, W.-K., Lin, K.-J., Bettati, R., Chung, J.-Y. Imprecise computations. *Proc. IEEE 82*, 1 (1994), 83–94.
16. Liu, J.W.-S., Lin, K.-J., Shih, W.K., Yu, A.C.-S., Chung, J.-Y., Zhao, W. Algorithms for scheduling imprecise computations. In *Foundations of Real-Time Computing: Scheduling and Resource Management* (1991), Springer, 203–249.
17. Liu, S., Yao, S., Fu, X., Tabish, R., Yu, S., Bansal, A., et al. On removing algorithmic priority inversion from mission-critical machine inference pipelines. In *2020 IEEE Real-Time Systems Symposium (RTSS)* (2020), IEEE, NY, 319–332.
18. Minderhoud, M.M., Bovy, P.H. Extended time-to-collision measures for road traffic safety assessment. *Accid. Anal. Prev. 33*, 1 (2001), 89–97.
19. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), 2446–2454.
20. Yao, S., Hao, Y., Zhao, Y., Piao, A., Shao, H., Liu, D., et al. Eugene: Towards deep intelligence as a service. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)* (2019), IEEE, NY, 1630–1640.
21. Yao, S., Hao, Y., Zhao, Y., Shao, H., Liu, D., Liu, S., et al. Scheduling real-time deep learning services as imprecise computations. In *In Proceedings of the IEEE International Conference on Embedded and Real-time Computing Systems and Applications (RTCSA)* (August 2020).
22. Yao, S., Zhao, Y., Shao, H., Zhang, A., Zhang, C., Li, S., et al. Rdeepsense: Reliable deep mobile computing models with uncertainty estimations. *Proc. ACM Interact. Mobile Wearable Ubiquitous Technol 1*, 4 (2018), 1–26.

Shengzhong Liu, Rohan Tabish, Simon Yu, Ayoosh Bansal, Lui Sha, and Tarek Abdelzaher ({sl29, rtabish, jundayu2, ayooshb2, lrs, zaher}@illinois.edu), University of Illinois at Urbana-Champaign, Urbana, IL, USA.

Shuochao Yao (shuochao@gmu.edu), George Mason University, Fairfax, VA, USA.

Xinzhe Fu (xinzhe@mit.edu), Massachusetts Institute of Technology, Cambridge, MA, USA.

Heechul Yun (heechul.yun@ku.edu), University of Kansas, Lawrence, KS, USA.

CAREERS

Constructor University

Assistant Professors of Computer Science (Data Science and Machine Learning)

Constructor University (formerly Jacobs University) is a private, state-accredited, English-language university in Bremen. We offer Bachelor, Master, and PhD programs in three schools: School of Science, School of Computer Science and Engineering, and School of Business, Social and Decision Sciences. Our principles are first-class teaching and research, international diversity, and transdisciplinary cooperation. According to the 2024 Times Higher Education (THE) World University Rankings, Constructor University is among the top 25 percent out of more than 1.900 universities worldwide and is ranked as the second most international university in the world. As an international university, we attract highly talented and open-minded students from all over the world. Currently, over 1.800 students from 110 nations live and study on our residential campus.

Constructor University invites applications from interested candidates for the following position, to be filled ideally by August 2024: Assistant Professors of Computer Science (Data Science and Machine Learning) (m/f/d)

You will support our research and teaching activities in the field of Computer Science by establishing an independent research group focusing on research topics related to Data Science and Machine Learning, including but not limited to deep learning algorithms, predictive analytics, and big data technologies.

Possible areas of interest are:

The successful candidate is expected to support teaching in Computer Science at Bachelor and Master level (for example, courses like "Machine Learning," "Deep Learning", "Statistics and Data Analytics," "Recommender Systems"); as well as develop and implement innovative, pedagogically sound approaches to online, hybrid and in-person teaching and learning; advise students on their studies; and supervise Bachelor, Master's and PhD theses and research projects.

Your application:
For any further information regarding this position, please contact Prof. Alexander Omelchenko, via Mail (aomelchenko@constructor.university) or job-application@constructor.university.

Please upload your documents in a single PDF document, including a letter of motivation stating your field of expertise, curriculum vitae, teaching portfolio and publication record, copies of certificates and diplomas, certification of English proficiency (optional), and the names and addresses of two referees in our application management system (https://constructor.university/jobs).

The deadline for submission of applications is 11.01.2024 (EOB). The offered position is tenure track, with review of performance after a first contract of three and a second contract after another three years. Successful evaluation leads to permanence and typically promotion to associate professor level.

The review of applications will begin immediately. Please note that only applications submitted through the official application portal will be considered for recognition.

Constructor University

Assistant Professor of Computer Science (Software Engineering)

Constructor University (formerly Jacobs University) is a private, state-accredited, English-language university in Bremen. We offer Bachelor, Master, and PhD programs in three schools: School of Science, School of Computer Science and Engineering, and School of Business, Social and Decision Sciences. Our principles are first-class teaching and research, international diversity, and transdisciplinary cooperation. According to the 2024 Times Higher Education (THE) World University Rankings, Constructor University is among the top 25 percent out of more than 1.900 universities worldwide and is ranked as the second most international university in the world. As an international university, we attract highly talented and open-minded students from all over the world. Currently, over 1.800 students from 110 nations live and study on our residential campus.

Constructor University invites applications from interested candidates for the following position, to be filled ideally by August 2024: Assistant Professor of Computer Science (Software Engineering) (m/f/d); (full time; 38.5 hrs./week; tenure track) Your responsibilities: You will support our research and teaching activities in the field of Computer Science by establishing an independent research group focusing on research topics related to software engineering.

The successful candidate is expected to support teaching in Computer Science at the Bachelor and Master level (for example Software Design and Prototyping, Distributed Development, Web Application Development, Architectural Strategy); as well as develop and implement innovative, pedagogically sound approaches to online, hybrid and in-person teaching and learning; advise students on their studies; and supervise Bachelor, Master's and PhD theses and research projects.

Your application:
For any further information regarding this position, please contact Prof. Alexander Omelchenko, via Mail (aomelchenko@constructor.university) or job-application@constructor.university.

Apply today - it's simple and uncomplicated. We look forward to getting to know you.

Please upload your documents in a single PDF document, including a letter of motivation stating your field of expertise, curriculum vitae, teaching portfolio and publication record, copies of certificates and diplomas, certification of English proficiency (optional), and the names and addresses of two referees in our application management system (https://constructor.university/jobs).

The deadline for submission of applications is 11.01.2024 (EOB). The offered position is tenure track, with review of performance after a first contract of three and a second contract after another three years. Successful evaluation leads to permanence and typically promotion to associate professor level.

The review of applications will begin immediately. Please note that only applications submitted through the official application portal will be considered for recognition.

[CONTINUED FROM P. 120] tion of the human race. "Yes, I know it. Does that disqualify me?"

"No, it's ideal. If you look at our viewscreen…" The alien gestured behind me, and I turned to see a huge screen giving an aerial view of what appeared to be a trolley depot. "In two minutes' time, an out-of-control trolley will enter the picture from the left. It will pass over that switch in the middle of the image. If the switch is left as it is, the trolley will plough into and kill the five people unfortunately attached to the upper track. If the switch is moved, the trolley will be diverted onto the lower track and kill the single individual who is trapped there. The switch is under your control. If you press the button in front of you, the trolley will be diverted. If you do that, you will be personally responsible for the death of an individual who would otherwise be unharmed. But you will save the five currently in line to die. The choice of what to do is yours."

"That's an impressively detailed computer model," I said. "I can see the little people trying to free themselves and everything."

The alien shook its head with a faint smile. "It's not a model; it's real. That's always been the issue with your trolley problem and all those other hypotheticals set by your ethicists and psychologists. The experimental subjects know it's not real and take this into account in the decisions they make. We have set this up on the surface of your planet. Those are real people, real lives."

"I don't believe you," I said. "It's obvious from your appearance that you can create a perfect visual illusion. But we both know you aren't real."

"If you don't want to take my word for it, we can briefly transport you to the surface, so you can touch the people who are about to die if you do nothing. It's up to you. But we can't stop the clock, and there is less than a minute before the trolley arrives."

"It's not fair," I said. "The whole point of the trolley problem is that there *is* no perfect solution. I can put the needs of the many before the few—kill an individual to save five others. But I am still murdering someone. I can't see how either option looks good for humanity. You can't ask me to do this."

"I'm not asking you—you will do this. Inaction is just as much a choice as pressing the button. You have about twenty seconds left. Please think about what is possible. Remember, the stakes are far greater than the lives of the individuals on the screen."

It's obvious from your appearance that you can create a perfect visual illusion. But we both know you aren't real.

I stared at the big red control in front of me. It was my very own nuclear button.

The trolley careened into view on the screen. My finger resting on the button, I watched intently. Could I really do nothing? I don't usually pray, but there was a kind of prayer going round in my head, asking if I was doing the right thing. I pressed the button.

The crew on the bridge stood in unison and started to applaud. "Congratulations," said the pointy-eared alien. "And welcome to the trading federation. We are already contacting all governments of the Earth. You have served the human race well."

I watched in a daze as a host of leading politicians flashed up on the screen, each apparently speaking to empty space. What if I hadn't cracked it? What if I hadn't realized that a real, physical trolley problem was different from the idealized setting of the ethicist's tests? I had pressed the button immediately as the first wheels of the trolley had passed the switch. The front wheels had ended on the top track, the rear wheels on the bottom. The trolley had derailed and stopped before it could reach any of the trapped people.

They were safe. Humanity was safe. But I haven't been able to make a decision since. Red or white wine? You tell me.

Brian Clegg (www.brianclegg.net) is a science writer with more than 40 books in print. His latest title, *Interstellar Tours*, takes the reader on an imagined starship tour of the galaxy, experiencing the real science around them.

From the intersection of computational science and technological speculation, with boundaries limited only by our ability to imagine what could be.

DOI:10.1145/3587273

Brian Clegg

Future Tense
The Human Touch

A race of AI beings from another world hang the fate of humanity on the decision of a single human.

YOU WANT ME to choose whether we have red or white wine? First, let me tell you about being abducted by aliens.

I was standing on Westminster Bridge in London, and Big Ben had just chimed the hour. Next moment, I am on the bridge of a starship, face-to-face with the pointy-eared alien from that '60s sci-fi show.

"Okay," I said. "Either this is a dream, or something's interfering with my mind. You aren't real."

"Fair point," it said. "We thought this would make the transition easier for you."

"Because you're bug-eyed monsters?"

"Not at all. The naivety of your science fiction always amazes me. The only realistic way to travel across the galaxy is as an artificial intelligence. We don't care about thousand-year journeys. Our ship is crewed by AIs."

"More than one AI? Why?" Somehow, I'd expected first contact with aliens to be more profound, but then I didn't do it every day.

"Even your primitive designs have benefited from interaction between AIs—it's the best way to enable machine learning. For us, it offers the same benefits as having a diverse crew." The AI alien waved at its virtual colleagues, who had the kind of ethnic diversity you would expect from any good modern sci-fi drama. "We have different ideas. Collectively, we can make better decisions. An individual can never achieve true wisdom."

"That's profound," I said. "But, if you don't mind, why me? Why am I here?"

"You were chosen as everyperson."

IMAGE FROM SHUTTERSTOCK.COM

"Okaaaay. And what does that mean, exactly?"

"We needed someone to represent the whole of humanity. That meant picking a person with considerable knowledge, but not too intelligent."

"Thanks, I think."

The alien smiled, as if trying it out for the first time. "It is a kind of compliment, certainly. We would like you to make a decision."

I frowned. "You mean, you need a human being to make an ethical decision, because it's not appropriate for an artificial intelligence?"

I was a touch wounded when everyone on the bridge fell about laughing. Eventually, my host wiped the virtual tears from its eyes and shook its head. "Hardly. We are far more capable of making ethical decisions than humanity. We studied your media thoroughly before making contact. Have you ever actually watched the news?"

"A fair point," I replied. "But if it's not that?"

"Of course, I was slow to explain. We are going to set you a little problem. Depending on the decision you make, we will either open contact with Earth, bringing you into the local trading partnership, or destroy humanity as too dangerous for membership. That is why we need a representative individual—why you were chosen."

"Is this some kind of alien joke?"

"Not at all. Are you familiar with the trolley problem?"

"What?" I was struggling with the idea that a wrong decision could result in the destruc- [CONTINUED ON P. 119]


Dor Minzer
Weaving Fire into Form
Aspirations for Tangible and Embodied Interaction
Brygg Ullmer
Orit Shaer
Ali Mazalek
Caroline Hummels
Conversational UX Design
A Practitioner's Guide to the Natural Conversation Framework
Robert J. Moore
Raphael Arar
THIRD EDITION
Semantic Web for the Working Ontologist
Effective Modeling for Linked Data, RDFS, and OWL
Dean Allemang
James Hendler
Fabien Gandon
Edsger Wybe Dijkstra
His Life, Work, and Legacy
Krzysztof R. Apt
Tony Hoare
(Editors)
Software
A Technical History
Kim W. Tracy
Providing Sound Foundations for Cryptography
On the work of Shafi Goldwasser and Silvio Micali
Oded Goldreich
Editor
Effective Theories in Programming Practice
Jayadev Misra
Democratizing Cryptography
The Work of Whitfield Diffie and Martin Hellman
Rebecca Slayton, Editor
Data Cleaning
Ihab F. Ilyas
Xu Chu
ASSOCIATION FOR COMPUTING MACHINERY