COMMUNICATIONS OF THE ACM
CACM.ACM.ORG
02/2023 VOL.66 NO.02

# Extracting the Essential Simplicity of the Internet

Overcoming the Data Bottleneck

HPC Forecast: Cloudy and Uncertain

Programming's Premature Obituary

Computational Linguistics Finds Its Voice

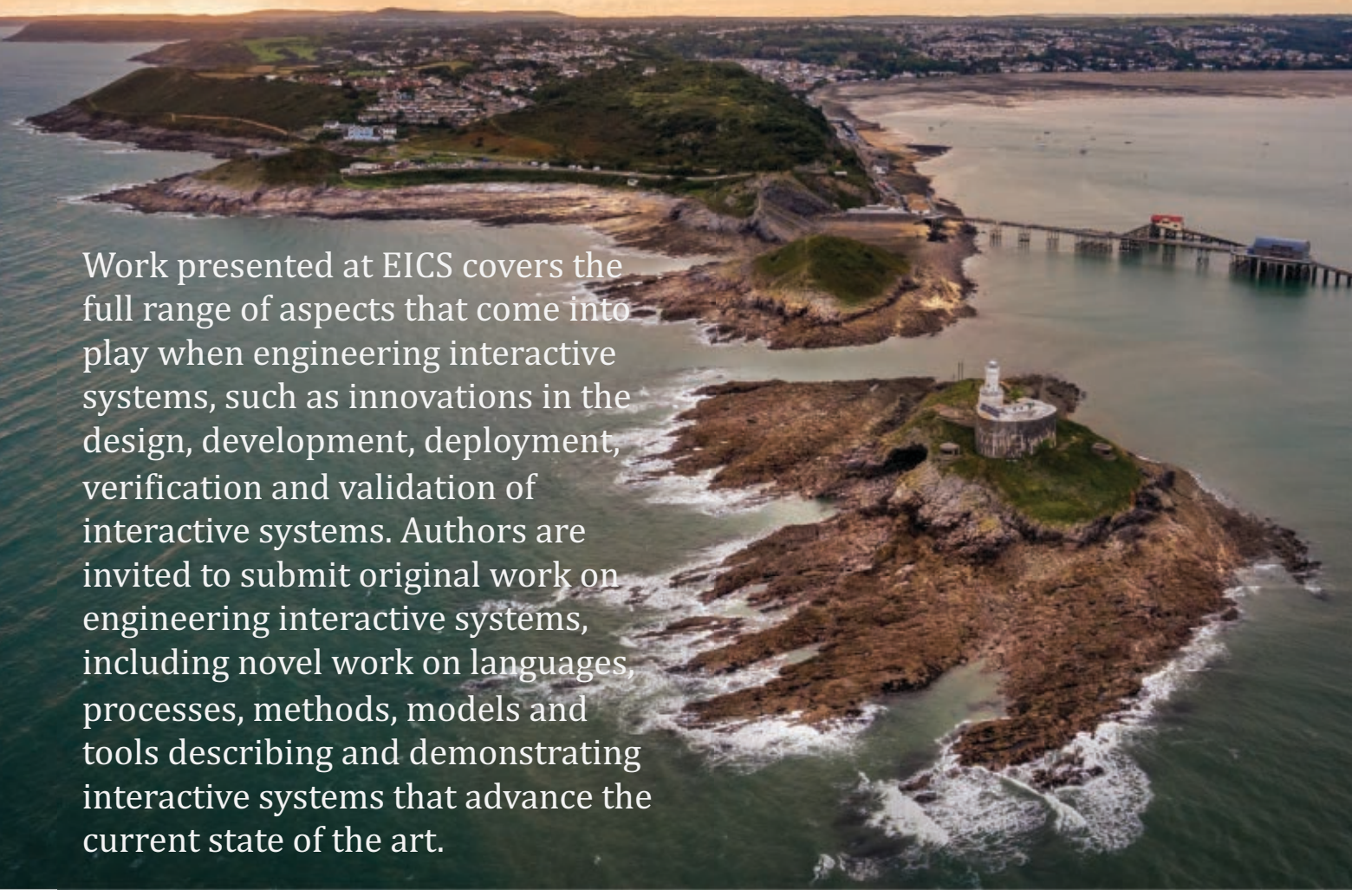
Association for Computing Machinery

# EICS 2023

## The 15th ACM SIGCHI Symposium on
# Engineering Interactive Computing Systems

Swansea, Wales, UK, 27 - 30 June 2023

eics.acm.org/2023/



Work presented at EICS covers the full range of aspects that come into play when engineering interactive systems, such as innovations in the design, development, deployment, verification and validation of interactive systems. Authors are invited to submit original work on engineering interactive systems, including novel work on languages, processes, methods, models and tools describing and demonstrating interactive systems that advance the current state of the art.

## Submission deadlines

Full papers
17 February 2023

Late Breaking Results
9 March 2023

Demonstrations and Posters
11 March 2023

Workshop proposals, Doctoral Consortium
17 March 2023

Sponsored By

Association for Computing Machinery

SIGCHI

ifip
IFIP WG 2.7/13.4

Swansea University
Prifysgol Abertawe

# ACM BOOKS

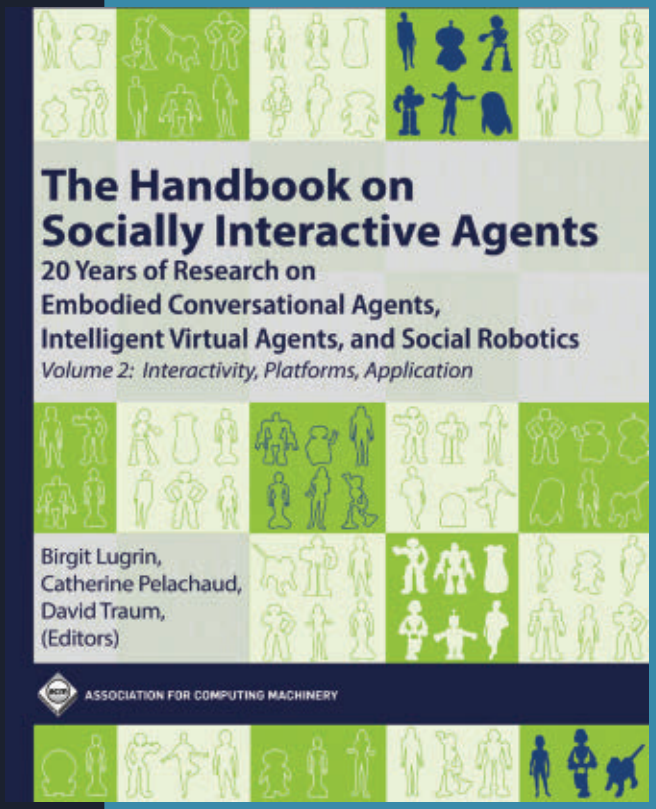
**Collection II**

*The Handbook on Socially Interactive Agents* provides a comprehensive overview of the research fields of Embodied Conversational Agents; Intelligent Virtual Agents; and Social Robotics. Socially Interactive Agents (SIAs); whether virtually or physically embodied; are autonomous agents that are able to perceive an environment including people or other agents; reason; decide how to interact; and express attitudes such as emotions; engagement; or empathy. They are capable of interacting with people and one another in a socially intelligent manner using multimodal communicative behaviors; with the goal to support humans in various domains.

Written by international experts in their respective fields; the book summarizes research in the many important research communities pertinent for SIAs; while discussing current challenges and future directions. The handbook provides easy access to modeling and studying SIAs for researchers and students; and aims at further bridging the gap between the research communities involved.

In two volumes; the book clearly structures the vast body of research. The first volume starts by introducing what is involved in SIAs research; in particular research methodologies and ethical implications of developing SIAs. It further examines research on appearance and behavior; focusing on multimodality. Finally; social cognition for SIAs is investigated using different theoretical models and phenomena such as theory of mind or pro-sociality. The second volume starts with perspectives on interaction; examined from different angles such as interaction in social space; group interaction; or long-term interaction. It also includes an extensive overview summarizing research and systems of human–agent platforms and of some of the major application areas of SIAs such as education; aging support; autism; and games.

**http://books.acm.org**
**http://store.morganclaypool.com/acm**




## The Handbook on Socially Interactive Agents

*20 Years of Research on Embodied Conversational Agents, Intelligent Virtual Agents, and Social Robotics, Volume 2: Interactivity, Platforms, Application*

**Birgit Lugrin, Catherine Pelachaud, David Traum (Editors)**

ISBN: 9781450398947
DOI: 10.1145/3563659

COMMUNICATIONS OF THE ACM

## Departments

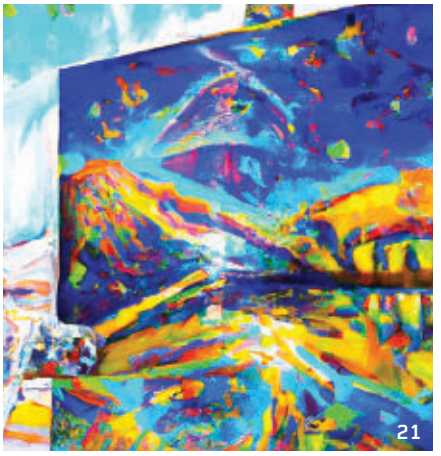


## Last Byte



Association for Computing Machinery
*Advancing Computing as a Science & Profession*

## News





## Viewpoints



IMAGE CREATED BY ANDRIJ BORYS ASSOCIATES WITH THE ASSISTANCE OF DALL·E 2

## Practice




Articles' development led by acmqueue
queue.acm.org

## Contributed Articles



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



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

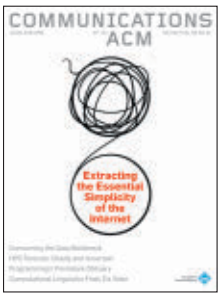


## Review Articles




## Research Highlights






**About the Cover:** Admiring and acknowledging the Internet's conceptually simple and brilliantly daring design is easy—once one steps back from the alphabet soup of complicated protocols. In this article, the authors attempt to extract the Internet's essential simplicity by motivating the fundamental Internet design choices from first principles without delving into all the ensuing complications. Cover image by Gwens Graphic Studio.

IMAGES BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES

# 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.

## ACM ADVERTISING DEPARTMENT

1601 Broadway, 10th Floor
New York, NY 10019-7434 USA
T (212) 626-0686
F (212) 869-0481

**Advertising Sales Account Manager**
Ilia Rodriguez
ilia.rodriguez@hq.acm.org

**Media Kit** acmmediasales@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

DOI:10.1145/3578649

Yannis Ioannidis

# To the Members of ACM

This is an update on the efforts of the ACM Executive Committee and Council to draft a longer-term strategic plan for our organization. At the core of our discussions have been critical and urgent issues raised by members during the Q&A process that ran during the recent ACM General Election, which also covered those I had presented in my candidacy statement. In my answers during that process, I promised to establish several Presidential Task Forces (PTFs) to investigate the issues raised and recommend concrete actions to address them. Considering the needs and priorities of ACM itself, the broader computing community, and society at large, the ACM Council has now endorsed 10 PTFs as the core dimensions of our strategic work. The strategic plan that will result from all PTFs together has been called *ACM 4.0*, an analogy to ACM entering the fourth quarter-century of its life. The main objectives of the PTFs are briefly described here.

Three PTFs address issues of ACM Membership:

***1. Membership Model:*** Exploring new membership models that create the sense of a unified large community but still convey the feeling of "home" to each member, offering different benefits that are meaningful to different constituencies. Identifying membership models that will entice all those who benefit from ACM to become members and will inspire those who are members to become volunteers.

***2. Globalization:*** Exploring different pathways toward geographic diversity at all possible levels: election and award candidates, volunteer leaders, board and committee members, and members in general. Investigating the formation of additional regional councils or other forms of governance structures that will collectively cover the world.

***3. Youthification:*** Exploring different pathways toward age diversity, with emphasis on the younger generation, at all appropriate levels: board and committee members and members in general. Identifying engaging activities to attract the pre-college generation to computing.

Two PTFs address issues of services to society at large:

***4. Code of Ethics:*** Establishing mechanisms to promulgate the ACM Code of Ethics, familiarize our community with the essentials of the Code, and educate it on ethical issues in general. Investigating how the Code, which applies to individuals, may become relevant to companies and other legal entities as well for the benefit of all.

***5. UN Sustainable Development Goals:*** Exploring appropriate mechanisms for ACM members to offer their computing technologies expertise in the context of multidisciplinary efforts toward achieving the 17 UN SDGs (and the Paris Agreement on Climate Change).

Two PTFs address issues of services to ACM members:

***6. Open Science:*** Establishing services to help researchers operate according to the Open Science paradigm (which includes Open Access as an important component) and the demands it brings, for example, reproducibility of experiments, transparency throughout the research life cycle, and new forms of scholarly communication and reviewing.

***7. Products and Services Portfolio:*** Taking a fresh look at the overall portfolio of ACM products and services, through the lens of the continuously evolving and diversified needs of different parts of the computing community, and examining issues such as the SIG structure, DL functionality, career development, mentoring, and others.

One PTF addresses issues of ACM finances:

***8. Financial Model:*** Designing a new financial model that will ensure the prosperity of ACM, in the context of a potentially new membership model and a renewed portfolio of products and services, in the fast-approaching era of ACM Open.

Two PTFs address issues of ACM internal processes:

***9. Regional Offices:*** Exploring the establishment of regional ACM offices around the world. Identifying the services to be offered by regional staff. Establishing a process for optimizing the choice of location and size of each office based on relevant parameters.

***10. Bylaws:*** Reexamining some fundamental aspects of our current bylaws that affect the effectiveness of ACM as a scientific professional society, including the lengths and limits of the terms served by leaders in key offices, and the structure of the Council with respect to its representativeness of all ACM member constituencies.

There are several bilateral connections and dependencies between these PTFs as well as between them, the currently established Councils and Boards, the SIGs, and staff, affecting their individual timing and mutual interactions. The goal is for all PTFs to have delivered their reports by early fall. Recruitment of PTF chairs and members is under way, proceeding both by invitation and through an open call for willing qualified members. I am asking all of you to step forward and contribute your time and expertise to this important strategic effort. ACM 4.0 will be what we all together make it to be! C

**Yannis Ioannidis** is President of ACM and professor of Informatics and Telecommunications at the National and Kapodistrian University of Athens and affiliated faculty at Athena Research Center, Greece.

# ACM Acknowledges the Companies Participating in ACM's Preferred Employer Program

The ACM Preferred Employer Program allows startups and small companies to provide ACM Professional Membership and ACM Digital Library access to their technical staffs at a preferred rate.

The following companies currently participate in ACM's Preferred Employer Program:

- Biteable
- Chameleon Consulting Group, LLC
- Circonus, Inc
- City of Portland - Bureau of Technology Services
- CompSci, LLC
- CrowdStrike
- Fastmail
- FullStory, Inc
- HashiCorp
- IntelliGenesis LLC
- Joyent, Inc.
- Lightelligence, Inc.
- Luminous Computing
- Micron Technology
- Nielsen Norman Group
- Riskfuel
- Surfshark
- tc1
- Van Andel Research Institute
- Virtek Vision International

Through the ACM Preferred Employer Program, each team member receives all the benefits of ACM Professional Membership, including *Communications of the ACM*, discounted registration fees for ACM SIG conferences, video-based online courses, digital books, skill-based learning paths, hands-on practice labs, coding sandboxes and more from leading providers Pluralsight Skills and Skillsoft Percipio.

**For more information and to apply:**
**www.acm.org/membership/preferred-employer**

acm Preferred Employer

acm Association for Computing Machinery

DOI:10.1145/3578891

Vinton G. Cerf

# On QR Codes and Safety

I had a meal at a relatively ordinary restaurant recently and was surprised when they brought the bill. It included a printed QR code. Out of curiosity, I took out my smartphone and used my camera app to "read" it. It took me to a website that asked for my credit card information. I followed the procedure on the website and paid my bill. Later, I realized I had no way of knowing where the QR code was going to take me or what else it might contain. There is no readily discernible information visible in a QR code (or many of its variations). You really have no way to tell whether it is potentially malicious. Originally intended for tracking purposes (for example, a package, a part in a manufacturing operation), this convenient mechanism for delivering digital information to a reader (for example, a camera feature) does not come with any human-readable way to ascertain safety.

Suddenly, QR codes on the sides of buildings, on magazine pages, or on websites all struck me as potentially threatening in the absence of any information to the contrary. Indeed, a casual web search for "QR codes threats" yields many hits raising this very issue. I read a few of these and began to wonder by what means users might be given more agency to determine the safety of information encoded in this fashion. A readable text associated with the QR code cannot really be trusted since the QR-coded information could be entirely distinct from the human readable part. Even human-readable URLs can range from misleading to malicious. Slight misspellings of domain names (for example, with lookalike Unicode characters) can take you to dangerous websites. Suitably composed malicious sites could automatically download malware as a result of activating a QR code (or clicking on a hyperlink).

Even if the QR-coded information is displayed to the user in readable form, there is no assurance it is safe or that its safety can be determined. A typical hyperlink could have a lot of unreadable but malicious content after the "?" in a URL, the safety of which is not obvious. So, the QR code safety question is, in a sense, comparable to the URL safety question which is, itself, a threat.

At the very least, maybe one wants QR readers to display the encoded material in as readable a form as possible before voluntarily following any encoded links? Perhaps there could be services that flag known-to-be-dangerous websites the QR reader could check? Perhaps a warning could be raised if an embedded domain name does not have a DNSSEC certificate? The browser used to interpret the input from following a URL might have a "sandbox" mode that would confine potential hazards.

**It seems apparent that the convenience of digital services comes with potential hazards against which we all need training and tools to avoid.**

In a recent essay,[a] I wrote about accountability and agency in the Internet, where I was concerned with identifying harmful actors and providing users with tools of protection. Perhaps safety-oriented QR code readers would fall into the latter category. It seems apparent that the convenience of digital services comes with potential hazards against which we all need training and tools to avoid. While it is apparent that people often give up privacy in exchange for convenience, we should not so lightly give up safety. This line of reasoning only further increases my growing belief we really do need the equivalent of "Internet driving courses" that conclude with the issuance of an "Internet driver's license." The license (or certificate) would simply be evidence that the student has been shown how to use the Internet more safely. We do not have Internet traffic cops to stop you from using the Internet in reckless ways, but perhaps we have an obligation to teach safety practices and provide tools that enhance them. C

a https://bit.ly/3vm1N7N

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

DOI:10.1145/3576929

# Neighborhood Watch

Vinton G. Cerf wonders "whether there is any possibility of establishing 'watcher networks'" in his October 2022 *Communications* "Cerf's Up" column. I must point out to all who have the same concern about "who will watch the watchers" that Philip K. Dick describes this problem in his story *The Minority Report* (see wikipedia https://bit.ly/2XlQcSA) where a group of three precogs (organic versions of AI) attempt to predict the future. It works often, but not always. So the question arises: How much authority are we willing to provide for AI and is the concept of three AIs working independently on the same problem a feasible solution. I agree with Cerf that we need to come up with a solution before the problem overwhelms us.

**Tom Jones,** Seattle, WA, USA

In the December 2022 *Communications*, there is a compelling column by Vinton G. Cerf, "On Truth and Belief," which exemplifies the growing worry about agreement, polarization, and the nature of truth. Even though I share the sentiments of Cerf, I believe his final assessment of the situation is counterproductive to his objectives, and by propagating this proposition, Cerf is diminishing the alternatives to the current state of affairs. As Cerf writes, "Perhaps critical thinking should be part of every educational curriculum from the earliest stages." There are two moments in this phrase worth discussing: criticism and education.

Let's start with the latter. A trend in contemporary politics is the attempt to solve societal issues by delegating them to the educational process. If there is any kind of habit, belief, skill, or competency that might be seen as fruitful for a society to come, the knee-jerk reaction is to add it to the educational curriculum. The solution is to make children acquainted with such an objective from early on, for example, to make the coming generation responsible for a political project of the current generation. This movement, however, represents a lack of responsibility of the current generation in taking the matter into their own hands. Perhaps, instead of proposing critical thinking in the educational curriculum, Cerf should propose to his readers the creation of reading groups, places for discussion, community centers, where the values and activities encompassed by his "critical thinking" can be exercised by people of his generation, our generation, thus taking the responsibility for fighting the erosion of agreement into our own hands.

More criticism, however, is possibly not what we need. Even though Cerf links criticism to agreement, he overlooks the fact that more and more criticism has become the very source of disagreement. Criticism, as the movement of finely analyzing arguments, their presupposition and their pragmatic consequences, even when well executed, has led toward debunking, skepticism, and suspicion. These are the same ingredients that brew conspiracy theories. The enlightened republic of letters is hardly returning in the age of digital media. If the option is to be a cynic critic, I believe agreement can only come in being noncritic, or perhaps post-critic. I posit the necessity of searching for new ways of engaging with discourse, truth and belief, instead of referring to the exhausted concept of criticism as a future possibility for agreement.

**More criticism, however, is possibly not what we need.**

In writing these final lines, Cerf has certainly embodied the opinions of many readers. These ideas, as I attempt to have briefly sketched, are not the optimal direction for the achievement of agreement and dialogue. By writing this letter, I hope to put these ideas to debate and perhaps ignite new thoughts on the considered matter.

**Luiz do Valle Miranda,** Kraków, Poland

**Author's response:**

*Thanks to Tom Jones for his reference to* The Minority Report*—an important reminder!*

*Luiz do Valle Miranda seems to have mistaken my use of "critical thinking" for "criticism." Critical thinking is closest to the practices of science in which one asks important questions about the sources and quality of information/data and also tests theories with falsifying experiments. Critical thinking is what keeps us from taking as true, assertions that are without basis. I think all ages benefit from this kind of thinking. Critical thinking is not criticism in my dictionary.*

**Vinton G. Cerf,** McLean, VA, USA

**Research and Practice**

The July 2022 issue of *Communications* has two advertisements on its first few pages. One raised the issue "Today's research driving tomorrow's technology" and the very next one spoke of a new book, *Theories of Programming*.

Reading further into the issue, I came to Jeanna Matthews' contribution challenging her readers to "consider reaching out with ideas for her column regarding Viewpoints." All of which caused me to consider the implications of all of that. An issue of great importance to me is the issue of the relevance of computing theory to computing practice. Does today's research really drive tomorrow's technology? Do the theories of programming make a contribution to practical programming? I do not know the answers to those questions, but I would sure like to.

It occurred to me that these are research questions, perhaps the most important research questions in the computing field. Does computing research/theory contribute to improvements in computing practice? If so, how? If not, why not? My own bias, as a dedicated practitioner, is that theory is all too often irrelevant to practice. But that is a

personal bias, and I would like to see that bias evaluated in a proper way.

The last time I saw any data, readers and members of *Communications* and ACM are predominantly practitioners. And the authors of the content of *Communications* are predominantly theorists. That tends to mean that *Communications* is written by one audience but intended for another. Therein, of course, lies a problem.

I think answering my questions is a legitimate goal of the ACM flagship publication, *Communications of the ACM*. The question is, of course, who among our theorist authors is willing to explore these questions.

**Robert L. Glass,** Nordland, WA, USA

**Editor-in-Chief's response:**

*Robert Glass raises several important questions in his letter. Let me respond with several of my own.*

*First, what is a CS theoretician? Aside from the actual field of theory, most computer scientists that I know build systems and run experiments. Many of them go even further and try to turn their ideas into tangible products by creating startups. The dividing line between research and practice in our field seems fuzzier and less well defined than in other scientific disciplines, perhaps because we create what we study. For example, the book Glass calls out (*Theories of Programming*) is the collected works of Tony Hoare, who invented both Quicksort and null pointers and produced seminal work on programming language semantics. Practitioner? Researcher? Theoretician?*

*Second, why are more practitioners not members of ACM (and readers of* Communications*)? To me, this is* the *existential question for ACM, given that its membership has held constant for more than a decade while the number of people with CS degrees has grown exponentially. Why have our students not joined ACM? Is there a better organization, magazine, or website for keeping up to date with our fast-changing field? Do practitioners not feel a need to stay up to date? Or, is ACM/*Communications *not relevant to the professional interests of practitioners?*

*These are all important questions, and I welcome the readers' perspectives on them.*

**James Larus,** Editor-in-Chief, *Communications of the ACM*
Lausanne, Switzerland

### Let's Build on Cybersecurity Technology Success

Moshe Y. Vardi's November 2022 *Communications* editorial "Accountability and Liability in Computing" about today's cyber-insecurity is to be applauded for distinguishing the success of "technical solutions" from "the market-failure issue." I have long made much the same distinction as reported in the May/June 2022 *IEEE Security & Privacy* blogpost "High Assurance in the Twenty-First Century", but with a very different conclusion regarding "technical solutions." Vardi asserts that 75 years into the computer age we still do not know how to build secure information systems. In contrast, I note that during the past 50 years the Security Kernel technical solution has "been a technology success, and a market failure."

What is the evidence of technology success? Five years ago, my November 2016 *Communications* Viewpoint "Cyber Defense Triad for Where Security Matters" pointed out "Security for cyber systems built without a trustworthy operating system (OS) is simply a scientific impossibility." The tech industry (like Microsoft) has asserted that this technology is "not able to cope with systems large enough to be useful." I responded that "this quite widely spread assertion has been repeatedly disproven by counterexamples from both real systems and research prototypes." Those included Security Kernels for SACDIN Minuteman missile control, NSA's BLACKER Internet cryptographic system, and commercial GEMSOS for primary IT for the Pentagon. The Viewpoint noted Security Kernel technology was applied for successful "deployments of highly secure systems and products, ranging from enterprise 'cloud technology' to general-purpose database management systems (DBMS) to secure authenticated Internet communications". With respect to security, it points out "At least a half-dozen security kernel-based operating systems have

**What is the evidence of technology success?**

been produced that ran for years (even decades) in the face of nation-state adversaries without a single reported security patch."

It is disappointing Vardi did not even acknowledge, let alone discuss, this contradicting evidence. I realize there are a couple of "sacred cows" that may also discourage authors from mentioning this technology: its demand for specific hardware support and its tension with open source ideology. Although segmentation and protection ring hardware were introduced for Multics security more than 50 years ago, Intel included them for security in their still ubiquitous x.86 architecture.

So why should we care that a highly respected expert source such as a *Communications* editorial did not mention a contradictory perspective? It is hardly surprising that an author emphasizes what is deemed supportive to a particular advocated direction—in this case "Accountability and Liability in Computing." The voice of such experts significantly influences the perception of "a lack of technical solution ... which disincentivizes those who may be able to fix serious security vulnerabilities from doing so." As I discuss in some detail in the *IEEE Security & Privacy* blogpost, it is most important that designers and manufacturers know they can build highly secure information systems on a Security Kernel-based trustworthy operating system, as has been repeatedly done in the past. This is affordable, commercially available technology with demonstrated good performance.

**Roger Schell,** Pacific Grove, CA, USA

**Author's response:**

*My* Communications *columns are one-page contributions. Under these constraints, there is no room for a scholarly review. Rather each column aims at making one point. In the column in question the point was the lack of accountability in computing due to lack of liability. I do not think that Schell and I are in disagreement. He argues that technical foundations already exist to build secure information systems, while I argued the slow progress in cybersecurity is due to market failure, which disincentivizes those who may be able to fix serious security vulnerabilities from doing so.*

**Moshe Y. Vardi,** Senior Editor, *Communications of the ACM*
Houston, TX, USA

**What Is Expectability in Prediction Modeling?**

In the November 2022 *Communications* India region special section "Hot Topics" contribution, "Toward Explainable Deep Learning," Vineeth Balasubramanian emphasized the necessity of achieving high levels in all three components of any algorithm: explainability, interpretability, and transparency. These components are even more crucial in health care compared to other domains. The computational models that are often widely used play important roles in decision support in, for example, liver allocation and cardiovascular risk assessment.

I would like to propose a fourth component denoted as "expectability." When a first machine learning prediction model is trained using a first subpopulation, and a second machine learning prediction model is trained using a second subpopulation, an expected behavior is observed if each model performs the best on the subpopulation it was trained on. For example, an unexpected behavior of the first machine learning model would be if the model yielded higher discrimination performance and/or closer-to-optimal calibration performance compared to the second model when deployed on the subpopulation the second model was trained on. Expectability would be achieved if any model trained on a certain subpopulation would always perform the best on similar subpopulations compared to any other models.

In analyses applied on the Explorys Life Sciences Dataset (an electronic medical record repository with over 21 million patients), we realized the widely used risk assessment tool in cardiology, the Pooled Cohort Equations, does not always behave as expected.[1] I led this research effort in collaboration with IBM Research and the Broad Institute of MIT and Harvard. The research revealed that one of the four equations, the one designed to predict outcomes on the Black men subpopulation, achieves the best calibration performance compared to the other three models when deployed on the three subpopulations it was not designed to function on. We explored the spectrum of possibilities to explain this unexpected behavior and

realized that several factors affecting calibration performance must be considered when developing risk assessment tools, which include variability between models such as the total number of coefficients, the size of the coefficients, and how close the coefficients are to zero in each model. Additional considerations must include skewness assessment, which relies on measures such as prevalence and predicted risk scores of patients in different risk ranges of the training and testing sets.

Any future prediction models must include assessments of expectability to evaluate expected behavior, using a variety of subpopulations and performance measures. High levels of expectability may even be most crucial because a model that performs well could be explainable, interpretable, and transparent; however, that model would still be suboptimal if it behaves unexpectedly.

**Any future prediction models must include assessments of expectability.**

**Reference**
1. Kartoun, U. et al. Assessing the robustness and internal consistency of the Pooled Cohort Equations (Poster). *American Medical Informatics Association 2022 Annual Symposium*; (Washington, D.C., Nov. 5–9, 2022).

**Uri Kartoun,** Cambridge, MA, USA

**Author's response:**

*Thank you for sharing this interesting perspective. The notion of "expectability" of machine/deep learning (ML/DL) models is, beyond doubt, critical. The DL research community has been viewing this from different perspectives such as model error calibration[3] (also see https://bit.ly/3PwdodG) conformal prediction (we have a book on this topic ourselves[1]), robustness,[2] out-of-distribution generalization,[4] transfer learning[6] and domain generalization.[5] However, your point of view is interestingly complementary to these explorations. Machine learning has always focused on the objective of "generalization performance," viz, the capability of the model to perform on data that it has not seen before. This has led to deliberate efforts on making a model do well on subpopulations that it did not see while training. It may be an interesting follow-up discussion to see when one would need expected calibration on subpopulations, and when one would need model generalization on other subpopulations. Either way, your point is an important one—and the closest topic I can think of within my limited knowledge is in terms of calibration/conformal prediction.*

*More generally speaking, while explainability (and related terms) are intended to* hedge *predictions and support the reasoning behind decision making, "expectability" is possibly a property of the model predictive performance itself. One could view these as different dimensions of the Quality-of-Service (QoS) levels an ML/DL model ought to support. When one views ML/DL as a paradigm of data-driven automation akin to its predecessor—software development—it is perhaps timely and critical for real-world ML/DL systems to have a rigorous life cycle with requirement specifications, development, evaluation, verification and quality control phases—especially in risk-sensitive and safety-critical applications.*

**References**
1. Balasubramanian, V. et al. (Eds). *Conformal Prediction for Reliable Machine Learning: Theory, Adaptations and Applications.* Elsevier Publishers, 2014.
2. Huang, X. et al. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review.* Aug. 2020.
3. Nixon, J. et al. Measuring Calibration in Deep Learning, CVPR Workshops 2021.
4. Shen, Z. et al. Towards Out-Of-Distribution Generalization: A Survey, arXiv:2108.13624
5. Zhou, K. et al. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* (Aug. 2022).
6. Zhuang, F. et al. A comprehensive survey on transfer learning. In *Proceedings of the IEEE* (Jan. 2021).

**Vineeth N. Balasubramanian,** Kandi, India

## Factoring Impact

Thanks for an interesting discussion of the pros and cons of virtual meetings, as we have suffered them for the past nearly three years as described in the November 2022 *Communications* "The Impact of Virtual Meetings." The News item lacks a discussion of how difficult virtual meetings are to the lip-reading deaf. I am such a person.

Often, low Internet speed, a speaker's slow Wi-Fi connection, or inefficient platform software make the refresh rate of the video of the speaker too low for lip-reading. Some platforms make the speaker's head a thumbnail, too small for lip-reading, in order to give maximum space to the speaker's slides. AI-generated captioning is offered as the solution to not being able to read lips. However, for all but the clearest, newscaster-like speakers, AI-generated captions are useless for understanding what a speaker is saying. So, I am often left guessing what a speaker is saying.

One virtual conference I attended used a platform that was low refresh rate by design—whose motto was "Audio is king in the world of online videos"—and that showed all slide-showing speakers' heads as thumbnails. The conference's solution was to provide me with AI-generated captioning. However, among all the lectures I attended, I was able to understand only one, a keynote, delivered by an unusually clear native English speaker, for whom the AI-generated captions happened to be useful.

Needless to say, I prefer in-person meetings and conferences.

I wonder how accessible virtual meetings are to the blind, with bitmapped images of slides that are unreadable by software that converts text to voice, and the fact that all voices now emerge from one loud speaker rather than from different places in a room.

**Daniel Berry,** Waterloo, ON, Canada

**Author's response:**

*Thanks for the comment. You raise a great point. It seems like there is a lot of work to be done making virtual meetings accessible to all. The story looks closely at why companies may or may not want to make virtual work the standard. But managers should be also considering the quality and nature of virtual work, too, before mandating such policies.*

**Logan Kugler,** Tampa, FL, USA

*Communications* welcomes your opinion. To contribute a letter to the editor, please limit your comments to 500 words or less and send to letters@cacm.acm.org

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:10.1145/3575663 https://cacm.acm.org/blogs/blog-cacm

# What Is Data Science?

*Koby Mike and Orit Hazzan consider why multiple definitions are needed to pin down data science.*


**Why Is It Hard to Define Data Science?**
December 2, 2022
https://bit.ly/3B9tZxK

If you ask a group of data scientists what data science is, you would probably hear different definitions. Indeed, although many attempts have been made to define data science, such a definition has not yet been reached. One reason for the difficulty to reach a single, consensus definition for data science is its multifaceted nature: it can be described as a science, as a research paradigm, as a research method, as a discipline, as a workflow, and as a profession. One single definition just cannot capture the diverse essence of data science. In this blog, we attempt to present the essence of each of these perspectives.

## Data Science as a Science

Empirical science has been always about data. Kepler used data about the movement of the planets collected by Tycho Brahe to prove Copernicus' theory of the solar system. Was Kepler the first data scientist when he looked for patterns and models in raw data? While Kepler used data to achieve insights, data science today is more than an empirical science. That is, *data science views the data itself as a natural resource and deals with methods for extracting value out of this data.*[10] While science focuses both on understanding the world and on developing tools and methods to perform research, data science focuses on understanding data and developing tools and methods to perform research on *data.*[11]

## Data Science as a Research Paradigm

Data science also introduces a new scientific paradigm. The first scientific paradigm, established thousands of years ago, is *empirical science*, in which scientists describe natural phenomena. The second scientific paradigm, applied hundreds of years ago, is the *theoretical paradigm*, in which scientists build models of nature. The third scientific paradigm was introduced only several decades ago and is the *computational paradigm*, in which scientists simulate complex phenomena using algorithms and computers. The fourth scientific paradigm, according to Gray,[7] is *data exploration*, in which data is captured or simulated, and then analyzed by scientists to infer new scientific knowledge. Following Gray, the National Institute of Standards and Technology (NIST) claimed data science is the current evolution of the fourth paradigm and described data science as *"the conduct of data analysis as an empirical science, learning directly from data itself. This can take the form of collecting data followed by open-ended analysis without preconceived hypothesis (sometimes referred to as discovery or data exploration)."*[3] In fact, this perspective views data science as the application of the grounded theory paradigm[6] for quantitative research.

## Data Science as a Research Method

Data science integrates research tools and methods taken from statistics and computer science that can be used to conduct research in various application domains, such as social science and digital humanities.

Drug discovery is one area that illustrates how machine learning is applied as a research method that includes *"target validation, identification of prognostic biomarkers and analysis of digital pathology data in clinical trials."*[13]

Another example is the application of machine learning methods in social science research. In such research, complex human-generated data, such as posts on social networks, are used to map social phenomena. Grimmer et al.[8] reviewed current use of machine learning in social science research and stated, *"inclusion of machine learning in the social sciences requires us to rethink not only applications of machine learning methods but also best practices in the social scienc-*

*es ... [machine learning] is used to discover new concepts, measure the prevalence of those concepts, assess causal effects, and make predictions. The abundance of data and resources facilitates the move away from a deductive social science to a more sequential, interactive, and ultimately inductive approach to inference."* As can be seen, data science is viewed in this case as a research method that transforms the research process from deductive to inductive, in line with the perspective on data science as a research paradigm presented here.

## Data Science as a Discipline

Data science integrates knowledge and skills from several disciplines, namely computer science, mathematics, statistics, and an application domain. One way to present such a relationship is using a Venn diagram. Conway[4] was the first to propose a Venn diagram for data science as a discipline;, many other Venn diagrams were proposed for the discipline of data science following Conway.[12] Figure 1 shows our Venn diagram for data science.

Researchers recognize three levels of integration between two or more distinct disciplines: *multidisciplinarity*, *interdisciplinarity*, and *transdisciplinarity*.[1] Multidisciplinarity is the lowest level of integration. In multidisciplinary education, learners are expected to gain knowledge and understanding in each discipline separately. *Interdisciplinarity* represents a higher level of integration than multidisciplinarity. In interdisciplinary education, after learners gain basic knowledge and understanding in each discipline separately, they are expected to understand the interconnections between the disciplines and to be able to solve problems that require applying different knowledge and methods from each discipline. In *transdisciplinarity*, boundaries among several disciplines transcend to create a holistic approach.

The gradual transition of data science from multidisciplinarity to interdisciplinarity introduces challenges concerning the definition of data science as a discipline that encompass the different bodies of knowledge, traditions, and cultures that originate in the different fields. Data science may eventually become a transdisciplinary domain.

**Figure 1. The data science Venn diagram (the authors' version).**

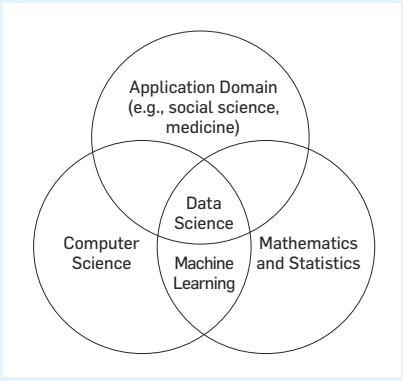
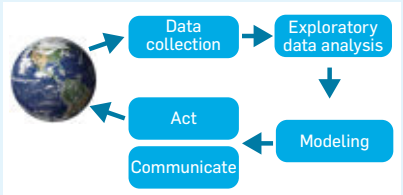



**Figure 2. Data science workflow (the authors' version[1]).**




The image of Earth was originally posted to Flickr by DonkeyHotey at https://flickr.com/photos/47422005@N04/5679642883. It was reviewed on December 4, 2020 by FlickreviewR 2 and was confirmed to be licensed under the terms of the cc-by-2.0.

## Data Science as a Workflow

The 2015 National Science Foundation (NSF) report, summarizing the NSF-sponsored workshop on data science education, introduced a definition of data science that reflects the perspective of data science as a workflow: *"Data science is a process, including all aspects of gathering, cleaning, organizing, analyzing, interpreting, and visualizing the facts represented by the raw data."*[2] Data science is indeed commonly presented as an iterative workflow for generating value and data-driven actions from data (see Figure 2).

## Data Science as a Profession

Irizarry[9] proposed that the term *data science* was coined in order to improve communication between human resource recruiters in the industry and work applicants. According to Irizarry, *"As the demand in [sic] employees capable of completing data-driven projects increased, the term data scientist quickly became particularly prominent because it helped recruiters specify the type of employee they wanted."*[9]

Accordingly, the definition of data science can be derived from the description of the profession of data scientist. And thus, for example, in their paper *"Data scientist: The sexiest job of the 21st century,"*[5] *Davenport and Patil describe the profession of data scientist as "a high-ranking professional with the training and curiosity to make discoveries in the world of big data ... More than anything, what data scientists do is make discoveries while swimming in data. It's their preferred method of navigating the world around them."*

## Conclusion

Data science is emerging fast and a single definition of it has not yet been reached. In this blog, we presented six facets of data science, each highlighting a different perspective of the field. This multifaceted nature of data science may partially explain the difficulties associated with the efforts to define it.

**References**

1. Alvargonzález, D. Multidisciplinarity, interdisciplinarity, transdisciplinarity, and the sciences. *International Studies in the Philosophy of Science, 25*(4), 2011, 387–403. https://doi.org/10.1080/02698595.2011.623366
2. Cassel, B. and Topi, H. *Strengthening data science education through collaboration: Workshop report 7-27-2016.* Arlington, VA, 2015.
3. Chang, W. L., Grady, N., et al. *Nist big data interoperability framework: Volume 1, big data definitions,* 2015.
4. Conway, D. The data science venn diagram. *Datist,* 2010. http://www.dataists.com/2010/09/the-data-science-venn-diagram/
5. Davenport, T. H. and Patil, D. Data scientist: The sexiest job of the 21st century. *Harvard Business Review, 90*(5), 2010, 70–76.
6. Glaser, B.G. and Strauss, A. L. The Discovery of Grounded Theory: Strategies for Qualitative Research. New York: Aldine de Gruyter, 1967.
7. Gray, J. *EScience – A transformed scientific method.* http://research.microsoft.com/en-us/um/people/gray/talks/NRC-CSTB_eScience.ppt, 2007f
8. Grimmer, J., Roberts, M. E., and Stewart, B. M. Machine learning for social science: An agnostic approach. *Annual Review of Political Science, 2021 24*, 395–419.
9. Irizarry, R. A. *The role of academia in data science education,* 2020.
10. Simberloff, D., Barish, B., Droegemeier, K., Etter, D., Fedoroff, N., Ford, K., Lanzerotti, L., Leshner, A., Lubchenco, J., Rossmann, M., et al. Long-lived digital data collections: Enabling research and education in the 21st century. *National Science Foundation,* 2005.
11. Skiena, S. S. *The data science design manual.* Springer, 2017.
12. Taylor, D. Battle of the Data Science Venn Diagrams. *KDnuggets.* https://www.kdnuggets.com/battle-of-the-data-science-venn-diagrams.html/, 2016.
13. Vamathevan, J., Clark, D., Czodrowski, P., Dunham, I., Ferran, E., Lee, G., Li, B., Madabhushi, A., Shah, P., Spitzer, M., et al. Applications of machine learning in drug discovery and development. *Nature Reviews Drug Discovery, 18*(6), 463–477, 2019.

**Koby Mike** is a Ph.D. graduate from the Technion's Department of Education in Science and Technology under the supervision of Professor Orit Hazzan. He is currently a post-doc at Bar-Ilan University. **Orit Hazzan** is a professor at the Technion's Department of Education in Science and Technology. Her research focuses on computer science, software engineering, and data science education. For additional details, see https://orithazzan.net.technion.ac.il/.

# SYSTOR 2023

## The 16th ACM International Systems and Storage Conference

## June 5-7, Haifa, Israel

Systor2023
June 5-7 Haifa, Israel

**Program Chairs:**

**Yossi Gilad** (Hebrew University, Jerusalem)
**Dejan Kostic** (KTH, Stockholm)

**Presentations on topics including:**

- Big Data infrastructure
- Cloud, edge, data center, and distributed systems
- Embedded and real-time systems
- Fault tolerance, reliability, and availability
- File and storage systems
- Networked, mobile, wireless, peer-to-peer, and sensor systems
- Operating systems, computer architecture, and their interactions
- Performance evaluation and workload characterization
- Runtime systems and compiler/programming-language support
- System deployment, usage, and experience
- System design or adaptation for emerging storage technologies
- System security and trust
- Systems for machine learning/machine learning for systems
- Virtualization and containers

**Visit our new web site!**

Science | DOI:10.1145/3575664 Don Monroe

# Post-Quantum Cryptography

*Cryptographers seek algorithms quantum computers cannot break.*

IMAGE BY ARTEMISDIANA

Exchanges of digital information have long been protected by public-key cryptography, which lets senders encrypt their data with confidence that only the intended recipient could decrypt and read it. The recipient can freely share their public key for encryption because deducing the private key for decryption would require an impractically large calculation, such as factoring the product of two very large primes.

In the 1990s, however, mathematician Peter Shor (then at Bell Labs and successor AT&T Research, now at the Massachusetts Institute of Technology (MIT)), showed quantum computers could do factoring and compute "discrete logarithms" exponentially faster, greatly stimulating research in quantum computing. Despite billions of dollars of investment and significant experimental progress, however, quantum computers are still too small and error-prone to pose a cryptographic threat—yet. If they do succeed at scale, though, both new data and encrypted archives could become vulnerable to snooping.

To address this possibility, the U.S. National Institute of Standards and Technology (NIST) has been evaluating proposals for "post-quantum cryptography" (PQC). Although it is now clear that quantum computers can solve some large problems much faster than classical computers ever could, the hope is that there are other cryptographic schemes that are forever beyond their easy reach.

## Hybrid Systems

The security risks are real, but they do not *directly* threaten the vast majority of encrypted data. "There is no quantum problem with bulk encryption," said security technologist Bruce Schneier. Most data is encrypted using symmetric algorithms, such as the Advanced Encryption Standard (AES), which uses the same key for decryption. As long as the sender and recipient can keep this shared key secret, this computationally efficient scheme works fine. A second quantum algo-

rithm developed in the 1990s by Lov Grover (then also at Bell Labs) can crack symmetric encryption faster than classical computing, but "the best you can do in speeding up this computation with a quantum computer ... is by a factor of a square root," Schneier said, so an attack can be defeated with modestly longer keys.

The challenge is securely establishing the shared key. Key distribution once used methods like a secure channel or physical exchange, but these are cumbersome for everyday use. (Quantum key distribution could detect any eavesdropping, but the demanding technology is unlikely to matter for commercial purposes.)

The situation changed with the 1976 work of Whitfield Diffie and Martin Hellman (who shared ACM's A.M. Turing Award in 2015), which described how two parties can share a secret without exposing it. The scheme exploits mathematical groups, such as elliptic curves, that support a multiplication operation. The parties agree on a starting point, then each applies a private multiplier and passes the result to the other, who then applies their own private multiplier. In this way they compute the same secret key, but the intermediate results can be sent in the clear without revealing it.

Although the multiplications are relatively easy, the scheme's security hinges on the exponential difficulty of reversing the calculation. A quantum computer, however, could do it much faster using Shor's algorithm, then use the result to decrypt bulk data encrypted with a symmetric algorithm.

### Not a Competition

NIST began preparing its program in 2011, said Lily Chen, a mathematician who is group manager of the cryptographic technology group, and began requesting PQC proposals in 2016. In addition to public-key protocols, the project solicited algorithms for digital signatures, which are used in key establishment, signing documents, and software source verification.

To conserve its own resources and the critical attention of the community, NIST narrowed 69 first-round submissions to 26 candidates for the second round, and then to seven finalists and eight alternative candidates for the third round. European collaborators are heavily represented in the proposals.

Selection assessed not only classical and quantum security, but also performance requirements such as processing power and key size, as well as side-channel resistance, Chen said. In addition, she said, "We wanted candidates from different categories," so the list included lattice-based, code-based, hash-based, multivariate-quadratic, and isogeny-based algorithms.

"Without the community we cannot do this work," Chen said, adding that open public review will be critical to the project's success. "There are not many people who understand the deep mathematics theory and also the cryptography to attack it."

In July 2022, NIST announced four finalists for their standards—CRYSTALS-Kyber for encryption and CRYSTALS-Dilithium, FALCON, and SPHINCS+ for digital signatures. NIST also chose several additional candidates for further consideration. The agency expects to release a formal standard in 2024.

### Classical Elimination

Despite the lengthy evaluation, some of the successful candidates have only recently been discovered to be vulnerable to classical attack. The multivariate public-key system Rainbow, for example, had been expected to join the ranks of digital signature algorithms. In March 2022, however, Rainbow was completely broken "over the weekend on a laptop," by Ward Buellens of IBM Research in Zurich, Switzerland.

Another proposal, called SIKE (for supersingular isogeny key encapsulation), had been in the further-study list for encryption. However, just weeks after the NIST announcement, Wouter Castryck and Thomas Decru, mathematicians at KU Leuven in Belgium, demonstrated how to crack it "in about one hour on a single core."

**"There are not many people who understand the deep mathematics theory and also the cryptography to attack it."**

The SIKE algorithm extends the Diffie-Hellman protocol. Rather than sender and receiver multiplying points on an elliptic curve by secret scalars, they perform a secret sequence of transformations (isogenies) on an entire elliptic curve. Unfortunately, clearly specifying this more-abstract process required exchanging some auxiliary data.

This is "a very nice trick actually, but also a trick that has been concerning already from the start" said Castryck, which led to it receiving only probationary approval for the fourth round. The extra information "was also the key ingredient in the attack," Castryck said. "I think NIST handled it correctly," he said, "but indeed, it makes you wonder whether you are standardizing things too quickly."

"It's a really good reminder to the community that these kinds of things can happen and do happen," said Peter Schwabe of the Max Planck Institute for Security and Privacy in Bochum, Germany, and Radboud University in Nijmegen, the Netherlands. "Somebody proposes something, and then many, many smart people look at it for a very long time and fail to break it," said Schwabe, who is a member of the CRYSTALS team. "Ideally, they fail to break it publicly, and describe how they failed breaking it, and then we gain trust in that system."

### Long Learning Process

However, "With the exception of hash-based signatures, the schemes that are being standardized now, and also the schemes that are still in round four of the competition, none of them have been studied to the same extent that, say, elliptic-curve crypto has that we're using today," Schwabe said.

Quantum attacks are even harder to assess, since "We don't really know yet what quantum computers can do," Schwabe said. Researchers have convincingly established that problems exist that quantum computers can

solve with polynomial resources, but which will always require exponential resources for classical computers. Google and others have demonstrated "quantum advantage" (originally called "quantum superiority") on selected problems even with today's research machines.

The goal of PQC, though, is practical algorithms that will assuredly require exponential resources even with a full-scale, error-corrected quantum computer. Indeed, a 2011 paper from Scott Aaronson (then at MIT, now at the University of Texas at Austin) and Andris Ambainis of the University of Latvia argued that exponential speedup could be achieved "only for certain 'structured' problems." For example, Shor's algorithm tackles factoring and discrete logarithms by identifying periodicity in the states of many quantum bits (qubits) at once. In April 2022, however, Takashi Yamakawa of the NTT Social Informatics Laboratories, and Mark Zhandry of NTT Research and Princeton University, conjectured that "structure" is not actually needed for exponential speedup for search problems (unlike the decision problems discussed by Aaronson and Ambainis). This represents the first major expansion of the scope of efficient quantum algorithms since the 1990s.

"It doesn't immediately say anything about whether there's classical attacks or quantum attacks on anything at this stage," Zhandry cautioned, but "It shows that there are potentially more problems where there is a quantum advantage relative to classical algorithms than there were previously." More importantly he said, even if an algorithm's security against classical attacks has been demonstrated, "Things will not be fine, necessarily," for quantum attacks.

## Establishing Trust

Schwabe noted there was "overwhelming evidence" the U.S. National Security Agency (NSA) had subverted protocols to retain some access for themselves, but with NIST's earlier competitions involving AES and Secure Hash Algorithm 3 (SHA-3), "They gained a really good reputation," he said. Today, "I would find it extremely hard to imagine that in any of the schemes there would be the kind of back door where only the NSA can decrypt," or that they would promote weakened protocols.

**"I would find it extremely hard to imagine that in any of the schemes there would be the kind of back door where only the NSA can decrypt."**

"NIST needs transparency if their standard is going to be used at all," Schneier agreed. "I think NIST is a trustable honest broker." However, "because you see things being broken left and right," he emphasized that future systems should be agile, letting users easily swap out broken protocols in the same way they might change the lock on a front door. C


### Further Reading

National Institute of Standards and Technology Computer Security Resource Center, Selected Algorithms for NIST Program on Post-Quantum Cryptography, https://bit.ly/3SpCyvg

*Shor, P.W.*
Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, *SIAM J.Sci.Statist. Comput.* 26 (1997) 1484. arxiv.org/abs/quant-ph/9508027

*Grover, Lov K.*
A fast quantum mechanical algorithm for database search, *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing.* STOC '96. Association for Computing Machinery: 212–219. arXiv:quant-ph/9605043

*Aaronson, S. and Ambainis, A.*
The Need for Structure in Quantum Speedups, *Proceedings of ICS (Innovations in Computer Science) 2011.* arxiv.org/abs/0911.0996.

*Yamakawa, T. and Zhandry, M.*
Verifiable Quantum Advantage without Structure, (2022). arxiv.org/abs/2204.02063


**Don Monroe** is a science and technology writer based in Boston, MA, USA.



# ACM Member News

### PIONEERING AFFECTIVE COMPUTING TECHNOLOGIES

**"As an electrical engineer, I wanted to know how computers worked," says Rosalind Picard, a professor of media arts and sciences at the Massachusetts Institute of Technology (MIT). "I wanted to explain it down to the electrons," she recalls, adding that this became her focus.**

**Picard earned her bachelor's degree in electrical engineering from the Georgia Institute of Technology, and her master's and doctorate degrees, both in electrical engineering and computer science, from MIT.**

**She worked as a member of the technical staff at AT&T Bell Laboratories in Holmdel, NJ, throughout graduate school. On completing her doctorate, Picard joined the faculty of MIT's Media Lab, where she has remained.**

**Picard's current research focuses on giving computers the skills of emotional intelligence in order to improve people's lives. She calls this affective computing, and she has pioneered the field.**

**More specifically, she says, "Today, I focus on AI (artificial intelligence) for digital health. It still has affective computing in it, but it's about helping people understand and manage their stress and moods."**

**Picard cofounded two companies that have commercialized affective computing technologies: Empatica, which markets wearable seizure alerting solutions, and Affectiva, which produces AI software that can understand human emotions and cognitive states by analyzing facial and vocal expressions.**

**In the future, Picard aims use computation and AI to help prevent certain health issues by revealing causal influences on different kinds of mental illnesses and health problems. "This will help people, in a personalized machine learning way, to know what they need to do to prevent those problems from happening," she says.**

***—John Delaney***

Technology | DOI:10.1145/3575666 Samuel Greengard

# Computational Linguistics Finds Its Voice

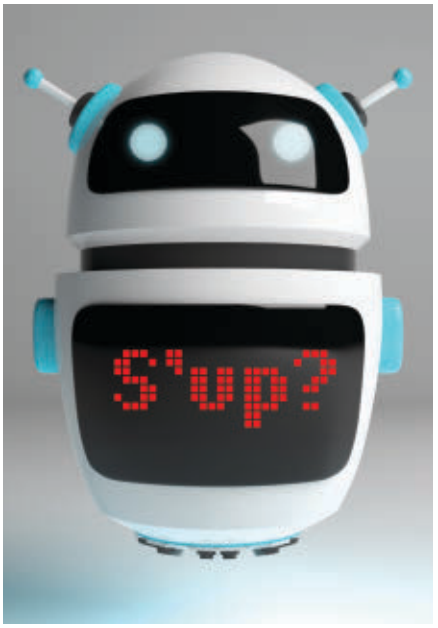
*Advances in artificial intelligence permit computers to converse with humans in seemingly realistic ways.*

Whether computers can actually "think" and "feel" is a question that has long fascinated society. Alan M. Turing introduced a test for gauging machine intelligence as early as 1950. Movies such as *2001: A Space Odyssey* and *Star Wars* have only served to fuel these thoughts, but while the concept was once confined to science fiction, it is rapidly emerging as a serious topic of discussion.

Thanks to enormous advances in artificial intelligence (AI)—and particularly deep learning—computers can now converse with humans in seemingly realistic ways. In a few cases, the dialog has become so convincing that people have deemed machines sentient. A recent example involves former Google data scientist Blake Lemoine, who published human-to-machine discussions with an AI system called LaMDA.[a]

Lemoine's declaration received ample press attention—along with a robust backlash from the computer science and artificial intelligence communities. For example, Stanford University researcher John Etchemendy, co-director of the Stanford Institute for Human-Centered AI (HAI) wrote: "LaMDA is not sentient for the simple reason that it does not have the physiology to have sensations and feelings."[b]

Yet, the mere fact that a computer could convince people it is sentient is not something to dismiss. Computational linguistics is advancing at a furious rate and natural language AI is taking shape. Startups such as Google's




LaMDA[c] and OpenAI's GPT-3[d] offer a glimpse at what is on the AI horizon. Already, machines are writing basic news articles[e] and tackling highly specialized chat functions. Future capabilities could revolutionize the way we go about our daily lives.

Avoiding the inevitable hype is critical. "There is this mad rush to claim that human-level AI is happening now," says Marjorie McShane, a professor of cognitive science at Rensselaer Polytechnic Institute. "Meaning and truly explainable AI still require a lot of serious work, with an uncertain payoff." Adds Julia Hirschberg, Percy K. and Vida L. W. Hudson Professor of Computer Science at Columbia University, "Although computers can generate human-like speech and text, there are still many issues to be resolved" to reach actual human-level interaction.

## Taking a Machine at its Word

Machine consciousness is a topic that has long fascinated researchers, media, venture capitalists, and the public. The Turing Test—originally called *The Imitation Game*—may have been the first attempt to analyze natural speech interactions generated by machines, but the path to the present has been paved with a string of advancements that have periodically sparked questions about whether computers can actually think and feel.

For example, in 1964, Joseph Weizenbaum at the Massachusetts Institute of Technology's Artificial Intelligence Lab introduced a program called ELIZA. Using a then-sophisticated pattern matching system, it interacted with humans in a contextual manner, albeit superficially. The program used words to trigger scripts that loosely followed rules of conversation. While Weizenbaum introduced ELIZA to demonstrate the fallibility of machine-human interaction, some observers concluded that the system was sentient. In other words, they believed that it had human-like feelings.

That might seem absurd by today's standards. However, Lemoine reignited the long-simmering discussion when he published an interview with the natural language system LaMDA. The chatbot clearly demonstrated a convincing command of language. Lemoine then took things further by claiming the system was actually sentient. He explored the idea of seeking legal representation and personhood for LaMDA. Several months later, Google unceremoniously fired him.

"Today's systems are very entertaining to talk to, but they don't build any

IMAGERY FROM SHUTTERSTOCK.COM

a Is LaMDA Sentient?—an Interview, https://bit.ly/3hxz8t1

b Is Google's AI sentient? Stanford AI experts say that's 'pure clickbait', *The Stanford Daily*.

c LaMDA: our breakthrough conversation technology, https://blog.google/technology/ai/lamda/

d GPT-3 Powers the Next Generation of Apps, https://openai.com/blog/gpt-3-apps/

e Robo-journalism: computer-generated stories may be inevitable, but it's not all bad news, https://bit.ly/3Fqu9T1

real model of the world," says Gary Marcus, co-author of Rebooting AI and founder and CEO of Geometric Intelligence, a firm acquired by Uber. Adds McShane, "ML systems can now generate text that, in many cases, sounds natural to people—except, of course, when they fail—sometimes with eye-popping incongruities."

The problem, McShane says, is that when faced with natural-sounding text, "People can't help but assume that the entity that created it understands it, but this couldn't be further from the truth." While deep learning and machine learning have advanced remarkably, and systems can spit out natural-sounding text, "Generating meaning-free language has nothing to do with actual intelligence in machines. Fluency is irrelevant when there's no meaning behind the text. These systems have no idea what they are talking about."

In fact, one group of researchers, including former Google AI scientist Timnit Gebru (now with Black in AI), labeled today's AI speech systems "stochastic parrots" in a 2021 academic paper.[f] In 2020, a separate team of researchers from the University of Washington noted, "Pretrained neural language models (LMs) are prone to generating racist, sexist, or otherwise toxic language, which hinders their safe deployment."[g]

"Sentience is an enormous logical leap," argues Stuart M. Shieber, James O. Welch Jr. and Virginia B. Welch Professor of Computer Science at Harvard University. "It doesn't follow that even if the Turing Test is a reasonable, if controversial, test for attributing intelligence, that it also serves as a reasonable test for attributing consciousness or sentience. People are deceived by the appearance of realistic behaviors."

## Mannerisms of Speech

Sentience aside, computer linguistics and natural speech are beginning to take root in the real world. Chatbots are getting better at mimicking human interaction. Autocomplete technology, used by Gmail and other applications, is changing the way people compose messages. Meanwhile, AI companies are introducing robots capable of carrying on full-fledged conversations with humans. One such firm, *Embodied*, offers a talking doll called *Moxie* that aims to promote social and emotional skill development for children between the ages of 5 and 10.

**"Sentience is an enormous logical leap. People are deceived by the appearance of realistic behaviors."**

OpenAI, an organization created by Elon Musk, Greg Brockman, and other heavyweights in the artificial intelligence field, is pushing the boundaries of natural speech. In 2020, OpenAI introduced Generative Pre-Trained Transformer 3, or GPT-3. The giant neural net's natural language skills are nothing short of remarkable; it can write articles and aid in software development, answer research questions using findings from academic papers, and handle a variety of text-generation and classification tasks. Popular language app *Duolingo*, for example, uses GPT-3 for French grammar corrections.

GPT-3, housed within a supercomputer complex in Iowa, was built using about 700 gigabytes of data originating from Wikipedia, digitized books, and websites. It includes approximately 285,000 CPU cores. Yet, while OpenAI boasts remarkable writing ability by machine standards—it can generate movie scripts and compose nonfiction in the style of famous authors—it remains a work in progress. At times it generates glaring errors and, when fed certain types of input, it is prone to spit out hollow prose.

Meanwhile, Google, DeepMind, and Meta all have developed their own language learning models (LLMs). Others, including Microsoft, Apple, Amazon, IBM, Nvidia, and Baidu also are continuing to advance machine linguistics and natural speech to address various requirements, whether the language skills are built into a digital assistant, a search engine or an interface to operate machinery.

Several methods exist for training natural language models, typically using convolutional neural networks (CNNs).

One common approach relies on so-called *word vectors*, which attempt to represent the meaning of a word mathematically, in order to highlight relationships between words and phrases.

Another approach, Recognizing Textual Entailment (RTE), classifies the relationships of words and sentences to each other through the lens of entailment vs. contradiction vs. neutrality. For example, the premise "A book has pages" entails "pages have text," but contradicts "words and images in a book move," while remaining neutral to a statement like "all books are good." As the system runs through millions of word combinations, it learns how to present and deliver statements accurately in context.

The goal, of course, is to develop systems that allow humans to interact with machines on human terms. Over the coming years, this could translate into new social media models for the metaverse and far more advanced sales and support functions. An AI system might, for example, provide highly interactive instructions for how to assemble furniture or manage a retirement account. Rather than reading cryptic instructions and struggling with basic YouTube videos, a person would simply ask for help and receive step-by-step contextual information including text, videos, and animations.

Of course, building systems capable of such real-world conversations is difficult, for a couple of reasons. First, there's the inherent complexity of language and the nearly infinite number of word combinations possible. No amount of training can prepare a system for every conceptual possibility. Second, computers lack any ability to discern red flags. A problem, Hirschberg points out, is that typical language samples fed into LLM systems

f On the Dangers of Stochastic Parrots: Can Language Models Be Too Big, https://dl.acm.org/doi/10.1145/3442188.3445922

g Real Toxicity Prompts: Evaluating Neural Toxic Degeneration in Language Models," https://aclanthology.org/2020.findings-emnlp.301.pdf

lack "bad language" and "undesirable content."

In the real world, Hirschberg adds, this inherent complexity presents problems for "conversational agents and chatbots that must deal with a great many different topics appropriately, from shopping to travel to reservations to just answering user questions on a variety of topics."

### The Final Word

Although computational linguistics capabilities almost certainly will advance by an order of magnitude in the coming years, what is not clear is how this will impact society. OpenAI has made it a point to focus on ethical AI, even publishing a paper featuring ways to battle the adverse use of its LLM.[h] Gebru and a group of other prominent researchers have established an organization named DAIR, which stands for *Distributed Artificial Intelligence Research*. Others, including Google, DeepMind, and Meta, are actively studying various aspects of ethics.

Yet major questions remain, and it is not clear whether well-intentioned AI ethical frameworks or even extremely accurate systems will solve the underlying problems. It is no secret that today's LLM models largely remain black boxes. There is virtually no understanding of how language models work. This makes it difficult to guarantee they will not cause damage or be commandeered for nefarious purposes—such as manipulating human behavior.

McShane and others believe one possible solution is developing agents that orient around meaning. Such agents extract the significance of language utterances; they reason, learn, and make decisions based on meaning, and they generate language based on the meaning they want to convey. So-called *Language-Endowed Intelligent Agents* (LEIAs)[i] rely on knowledge-based methods and cognitive modeling while incorporating the results of machine learning when and where it is applicable. A primary benefit of orienting a language system around meaning is that these agents will be able to explain their behavior in ways that people can understand.

Yet, even if total model transparency is achieved, there are no guarantees a system will work exactly as billed. Even if a broader set of people design and manage the technology—and government regulation is in place—things could still go haywire. "We're often seduced into thinking that even well-intentioned systems are good, but we really don't understand the full impact of the technology. While the performance of these systems has improved, our structured understanding of their behavior hasn't commensurately improved," Shieber says.

Concludes Marcus, "A lot of people want to think that more data will solve the problem, but that's what they have been telling us for a decade with driverless cars. So far, this approach hasn't worked, and it isn't clear that it will ever work. Both in language and driving, there is a constant stream of 'edge cases'—things that programmers didn't anticipate, and we're not equipped to deal with." C

h Improving Language Model Behavior by Training on a Curated Dataset; https://openai.com/blog/improving-language-model-behavior/

i Linguistics for the Age of AI (*MIT Press*, 2022); https://direct.mit.edu/books/book/5042/Linguistics-for-the-Age-of-AI

### Further Reading

*Bender, E.M., Gebru, T., McMillan-Major, A., and Shmitchell, S.*
**On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? *FAccT '21: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, March 2021 Pages 610–623. https://doi.org/10.1145/3442188.3445922**

*Sheiber, S.*
**There Can Be No Turing-Test—Passing Memorizing Machines, *Michigan Publishing*, June 2014, Volume 14, No. 16, pp. 1-13. https://dash.harvard.edu/handle/1/11684156**

*Gehman, S., Gururangan, S., Sap, M., Choi, Y., and Smith, N.A.*
**Real Toxicity Prompts: Evaluating Neural Toxic Degeneration in Language Models. https://aclanthology.org/2020.findings-emnlp.301.pdf**

*McShane, M.*
**Linguistics for the Age of AI (MIT Press, 2022). https://direct.mit.edu/books/book/5042/Linguistics-for-the-Age-of-AI**

**Samuel Greengard** is an author and journalist based in West Linn, OR, USA.



Milestones

# ACM Honors 2022 Distinguished Members

**ACM has named 67 new Distinguished Members for their significant contributions. All were selected for work that has spurred innovation, enhanced computer science education, and moved the field forward.**

**"The ACM Distinguished Members program honors both accomplishment and commitment," said ACM president Yannis Ioannidis. "Each of these new 67 Distinguished Members have been selected for specific and impactful work, as well as their long-standing commitment to being a part of our professional association. As ACM celebrates its 75th anniversary this year, it is especially fitting to reflect on how our global membership has built our organization into what it is today."**

**The 2022 ACM Distinguished Members work at leading universities, corporations, and research institutions in Australia, Canada, China, Finland, France, Germany, Greece, India, Italy, Japan, the Netherlands, New Zealand, Singapore, Taiwan, the U.K., and the U.S.**

**ACM Distinguished Members are selected for their educational, engineering, and scientific contributions. This year's Distinguished Members made advancements in areas including algorithms, computer science education, cybersecurity, data management, energy efficient computer architecture, information retrieval, healthcare information technology, knowledge graph and semantic analysis, mobile computing, and software engineering, among many others.**

**A candidate for Distinguished Member must have at least 15 years of professional experience in computing and five years of professional ACM membership in the last 10 years, and must have achieved a significant level of accomplishment or made a significant impact in the field of computing. They also are expected to have served as a mentor and role model by guiding technical career development and contributing to the field beyond the norm.**

Society | DOI:10.1145/3575665 Keith Kirkpatrick

# Can AI Demonstrate Creativity?

*When fed a sufficient amount of training data, artificial intelligence techniques can be used to generate new ideas in several different ways. Is that creativity?*

Creativity has been defined as the use of the imagination or original ideas, especially in the production of an artistic work. While the source of the development of those ideas can be debated—does creativity spring from the heart, the brain, the soul, or one's experiences—it has been largely accepted that humans alone possess the capability to truly create.

The emergence of computers and artificial intelligence (AI) has led to systems that, fed a sufficient amount of training data, can mimic the output of a creative writer, artist, or musician, thereby encroaching on humans' monopoly on the creative process. Artificial intelligence techniques can be used to create new ideas in a few different ways, such as producing unique combinations of familiar ideas, creating new works based on the attributes of previous works, and by offering new ideas based on combinations of attributes and ideas that humans may not have thought of during the creation of a new work.

A notable example of the power of AI to generate a so-called "creative" work was demonstrated in 2016 when the IBM Watson AI platform was used to create a movie trailer for 20th Century Fox's horror film, *Morgan*. The first example of a trailer created solely by AI, Watson was used to analyze the visuals, sound, and composition of hundreds of existing horror film trailers, and then it selected scenes from the completed *Morgan* movie for editors to patch together into a trailer. The use of AI to comb through scenes to create a trailer in the style of other horror movies helped to reduce the amount of time editors needed to spend on the project from a week down to a single day.

## How AI Can Mimic Creative Works

The process for using AI to generate cre-

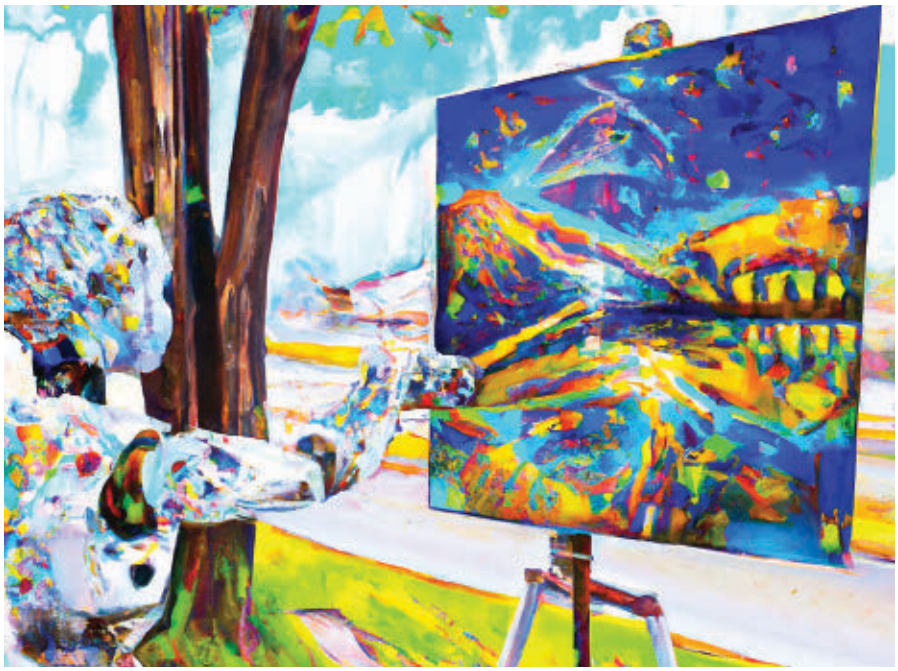


**Image created by DALL•E 2 when prompted, "AI envisioned as an artist creating beautiful art."**

**Image created by DALL•E 2 when prompted, "the most wildly creative image imaginable."**

ALL IMAGES CREATED BY ANDRIJ BORYS ASSOCIATES WITH THE ASSISTANCE OF DALL-E 2

ative content is largely based around the use of foundational models or generative adversarial networks. These approaches utilize deep neural networks designed to mimic the ways in which the human brain learns by creating associations between specific elements that can be combined to create a finished work.

These neural networks are fed millions or billions of examples of a particular output (which could include images, sound samples, or text passages), which they subject to a sophisticated type of pattern matching to "learn" specific attributes, patterns, or cues. For example, algorithms that are used to create artwork in the style of impressionist artists would be shown works from Monet, Renoir, Manet, Degas, Cezanne, and Matisse, generally considered to be masters in this style of artwork. The neural network examines the works as patterns of pixels, and can be trained to identify the specific patterns that define the impressionist style. This creates a framework of knowledge that can be used to create a new work based on the learned parameters and attributes. The more "layers" or "depth" the model has, the more complex the resulting patterns and correlations can be.

Large AI companies such as OpenAI (which describes itself on its website as "a research and deployment company" whose mission is "to ensure that artificial general intelligence benefits all of humanity") have created applications such as DALL-E (referencing the artist Salvador Dali), which was announced in January 2021, and demonstrated that this approach could reproduce and recombine features from those existing images in new and aesthetically pleasing ways. The next version, DALL-E 2, released a year later, featured improvements to image quality and demonstrated that the system could reproduce different artistic styles.

**There is an inherent level of reality within works created by AI, because models are trained on existing works that incorporate aspects humans find pleasing.**

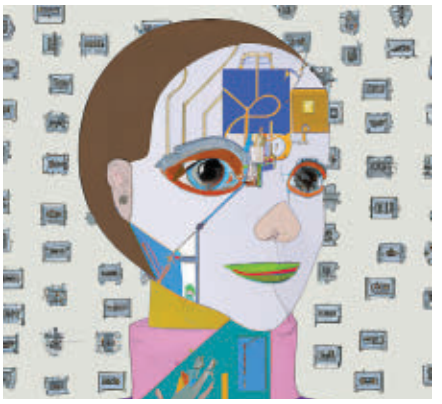


**Image created by DALL•E 2 when prompted, "AI creative self portrait."**

A similar approach can be taken with other types of creative works, such as music, where a neural network would be fed examples of music in order to learn specific patterns (such as common chord progressions, melodies, rhythms, and structure) used within a given genre, artist's repertoire, or other style data (for example, data distinguishing a waltz from a 12-bar blues or a jam-band-style improvisation).

This approach was used to create a song called "Break Free," which was composed by musician and YouTuber Taryn Southern. Southern utilized an open source platform called Amper Music (www.ampermusic.com) to create the basic stems, a grouped collection of audio sources grouped together as a single unit that will be combined with other stems during the mixing process. By choosing parameters that governed the song's structure, including the tempo, rhythm, type of instrumentation, and style, the AI platform created possibilities that Southern could choose from to create a song. Southern picked the possibilities she liked, and then arranged the pieces into a song structure to fit the lyrics she had written.

### Can AI Be Considered "Creative"?

Despite the ability of AI to produce creative outputs based on the attributes of existing works, the process is not the same as a human's creativity, which comes from a combination of real-world experience, emotion, and inspiration.

Indeed, these neural-network-driven systems lack real-world understanding of the world, and without the proper guardrails or parameters, can produce works that are nonsensical or strange. Further, because these outputs are based on the images, sound clips, or texts on which they are trained, they can reflect certain societal biases, such as creating artwork that renders, say, pilots as male, nurses as female, or song lyrics overrun with racial slurs or obscenities.

"Actual creativity is difficult to replicate. However, machine learning is an expert in finding patterns," says Wayne Butterfield, a partner of ISG Automation, a unit of global technology research and advisory firm ISG. "If these patterns are deemed as creativity, then, yes, computers can appear to be creative. The reality is for AI, each note, brush stroke, design, or other aspect of artistic expression is based on existing data, versus a true, original stroke of genius."

There is an inherent level of humanity within works created by AI, because models are trained on existing works that incorporate attributes humans find pleasing, such as certain combinations of colors, notes, or words. That said, "A computer is not a sentient being, and so it needs to have an input and then it will have an output," says Roger Firestien, a senior faculty member of the Center for Applied Imagination at the State University of New York College at Buffalo, a creativity consultant to Fortune 500 firms, and author of books detailing how creative ideas can be generated and applied to problems. "It seems like there's still always going to be some sort of human element there, at least at the genesis of using the technology."

## A Catalyst for Creativity

Perhaps the greatest change AI will impose on the creative arts is the expansion of the number of people or organizations that can generate and experiment with art, writing, and music. This can impact not only individual creators, but organizations that utilize creative works. For purely commercial applications, such as television scores or jingles, AI can be seen a method for improving the overall quality of a production, while reducing costs.

"Twenty years ago, television shows like A&E's *Biography* were still paying composers to score the episodes with original music," says Bob Higgins, a television producer and songwriter based in New York City. "That practice was replaced by blanket deals with music libraries offering cheaper, albeit generic, instrumental cuts for pennies. Now, it sounds like shows can have original music again, cuts created specifically for the show, but also done on the cheap by AI."

**"For AI, each note, brush stroke, design, or other aspect of artistic expression is based on existing data, versus a true, original stroke of genius."**

Even as AI improves, it is likely the technology will be used to augment, rather than replace, human creators. "Rather than put people out of work, using AI alongside their own skills enables even jingle writers, artists, etc., to do more, faster and potentially cheaper," Butterfield says. Further, he says, AI already has helped content creators scale their efforts in another domain: the development of realistic, feature-rich video games.

"We are already seeing AI-generated landscapes in video games doing the heavy lifting in our most immersive games," Butterfield says. "AI is enabling a bigger game to be built, which simply would be uneconomical if purely human computer programmers were used."

From a true artistic perspective, AI is likely to find use as a catalyst to spur more human creativity, much in the way recording studio technology has been used by musicians to expand how their final works were presented. Recording artist and guitarist Les Paul was a pioneer of using technologies such as multitrack recording and overdubbing to create now-classic recordings featuring his then-wife, Mary Ford, singing not only the melody of a song, but multipart harmonies.

Similarly, Higgins notes that the Beatles found new sounds on their groundbreaking *Revolver* album by playing the tape of a guitar solo in reverse. "It [was] music inspired by the technology used to create it," Higgins says. "Inspiration is a songwriter's bread and butter."

It is likely AI will become another tool for creators, rather than replacing the act of creation itself, which is usually considered a fulfilling and enjoyable process for artists, writers, and musicians. Firestien, a musician himself, says the process of using AI to come up with novel musical riffs or motifs could serve as a catalyst or starter for a composer suffering from writer's block. Indeed, Firestien notes, "Much of the joy of music is in the composing. Why would we want to teach a computer to compose when composing is so much fun?"

For the consumers of art, poetry, literature, music, or other creative content, enjoyment is often derived from the authentic, shared human experiences that are referenced to create that art. "It's in the person," Firestien says. "Can a computer fall in love? No. Can a computer be depressed? No. Can a computer go through the pandemic? No."

Technology is used when it is convenient and desirable for certain tasks. Despite inventions such as the telegraph, the telephone, email, and Web videoconferencing, people still desire to visit each other in person. Similarly, AI is likely to be used by artists, musicians, and other creators only to the extent it suits the needs of their creative process.

"Computers and AI can be good starters," Firestien says. "And maybe good finishers as well."

### Further Reading

*Schwabel, D.*
**How Taryn Southern used AI to Create an Album, *Forbes*, Sept. 26, 2017, https://bit.ly/3PNYPR3**

**DALL·E: Creating Images from Text, OpenAI, January 5, 2021, https://openai.com/blog/dall-e/**

*Brownlee, J.*
**An introduction to Generative Adversarial Networks (GANs), Machine Learning Mastery, https://bit.ly/3pNjV7J**

**What are Foundation Models?, IBM Research Blog, https://research.ibm.com/blog/what-are-foundation-models**

**Keith Kirkpatrick** is principal of 4K Research & Consulting, LLC, based in New York, NY, USA.

# ACM ON A MISSION TO SOLVE TOMORROW.

Dear Colleague,

I feel privileged to call ACM my professional home! Believing in the core ACM values of excellence, high ethics, and inclusivity, I joined as a graduate student and since then have been contributing in several ways to ACM's mission of "advancing the art, science, engineering, and application of computing."

For more than 75 years, ACM has played a leadership role in the computing field, keeping a finger on the pulse of the evolving needs of the community and expanding its extensive services to benefit computing researchers, industry professionals, educators, and students around the world.

More and more, ACM is taking steps to become the home of interdisciplinary areas that involve computing. ACM is helping to shape many computing-related interdisciplinary areas by forming strategic alliances with other scientific societies and welcoming members with multidisciplinary backgrounds.

ACM is also among the frontrunners in redefining scholarly communication and the entire research life cycle by embracing the principles of reproducibility and accountability and facilitating open science methods. ACM is actively transitioning to become a fully open access publisher within the next few years.

For many years, ACM has advised policy makers on cutting-edge technologies that may have significant consequences on society. Today, this work is more urgent than ever as threats on democracy, increasing inequalities, and the loss of privacy challenge us all. ACM continues to advocate for the application of technological innovation within clear ethical boundaries.

ACM has had tremendous impact on the profession by evolving continuously to better serve the needs of its diverse members, listening to all voices, and making innovative changes. This remains the course so that ACM can continue to fulfill its vision of being "the premier global computing society."

Joining ACM means you dare to be the best computing professional you can be. It means you believe in advancing the computing profession as a force for good. It means you act on your commitment to solving tomorrow's challenges. Come join your peers at ACM and let us all together invent a better tomorrow!

Sincerely,

Yannis Ioannidis
President
Association for Computing Machinery

Association for Computing Machinery

*Advancing Computing as a Science & Profession*

# SHAPE THE FUTURE OF COMPUTING.
# JOIN ACM TODAY.

www.acm.org/join/CAPP

## SELECT ONE MEMBERSHIP OPTION

### ACM PROFESSIONAL MEMBERSHIP:

- ❑ Professional Membership: $99 USD
- ❑ Professional Membership plus ACM Digital Library: $198 USD ($99 dues + $99 DL)

### ACM STUDENT MEMBERSHIP:

- ❑ Student Membership: $19 USD
- ❑ Student Membership plus ACM Digital Library: $42 USD
- ❑ Student Membership plus Print *CACM* Magazine: $42 USD
- ❑ Student Membership with ACM Digital Library plus Print *CACM* Magazine: $62 USD

❑ **Join ACM-W:** ACM-W supports, celebrates, and advocates internationally for the full engagement of women in computing. Membership in ACM-W is open to all ACM members and is free of charge.

## PAYMENT INFORMATION

Name

Mailing Address

City/State/Province

ZIP/Postal Code/Country

❑ Please do not release my postal address to third parties

Email Address

❑ Yes, please send me ACM Announcements via email
❑ No, please do not send me ACM Announcements via email

❑ AMEX ❑ VISA/MasterCard ❑ Check/money order

Credit Card #

Exp. Date

Signature

### Purposes of ACM

ACM is dedicated to:

1) Advancing the art, science, engineering, and application of information technology
2) Fostering the open interchange of information to serve both professionals and the public
3) Promoting the highest professional and ethics standards

By joining ACM, I agree to abide by ACM's Code of Ethics (www.acm.org/code-of-ethics) and ACM's Policy Against Harassment (www.acm.org/about-acm/policy-against-harassment).

I acknowledge ACM's Policy Against Harassment and agree that behavior such as the following will constitute grounds for actions against me:

- Abusive action directed at an individual, such as threats, intimidation, or bullying
- Racism, homophobia, or other behavior that discriminates against a group or class of people
- Sexual harassment of any kind, such as unwelcome sexual advances or words/actions of a sexual nature

# BE CREATIVE. STAY CONNECTED. KEEP INVENTING.

Association for Computing Machinery

ACM General Post Office
P.O. Box 30777
New York, NY 10087-0777

1-800-342-6626 (US & Canada)
1-212-626-0500 (Global)
Hours: 8:30AM - 4:30PM (US EST)

Fax: 212-944-1318
acmhelp@acm.org
acm.org/join/CAPP

DOI:10.1145/3576930

Barbara Ericson

▸ **Mark Guzdial,** Column Editor

# Education

# Four Ways to Add Active Learning to Computing Courses

*How active-learning techniques can benefit students in computing courses.*

Undergraduate computing classes typically deliver content through passive lectures and require students to write code from scratch. However, students do not always pay attention in lecture and writing code from scratch can be overwhelming for novice students. Students report feeling frustrated when they cannot figure out what is wrong with their code or wait hours to get help from instructional staff. Students from groups that have been historically marginalized are more at risk of failure in introductory courses since they tend to have less prior programming experience. How can we help students succeed in programming courses, especially those without prior programming experience?

Research on active learning in STEM has shown it improves learning, motivation, and pass rates over traditional lecture. In active learning students construct knowledge through discussion, problem solving, role play, and other methods. A meta-analysis found that students in traditional lecture STEM classes were 1.5 times more likely to fail than those in active-learning classrooms.[3] Active learning is particularly effective for students from historically marginalized groups. While there are many types of active learning, in this column I focus on four types: interactive ebooks; Peer Instruction (PI); mixed-up code (Parsons) problems and Process Oriented Guided Inquiry Learning (POGIL).

I have used these approaches in my lectures for several years in a data-oriented programming course, SI 206, which I teach in the School of Information at the University of Michigan. It is the second required programming course for our majors, but approximately 30%–40% of the students are from outside the School of Information. Many of the students are from computer science or engineering and know C++, but want to learn Python. Some of the students are business majors. There is a large variance in prior programming experience, familiarity with Python, and in the level of programming skill that students want to develop. The course enrollment has doubled in five years, from approximately 100 students in 2018 to approximately 200 students in winter 2023. Readings are assigned before lecture from a free and interactive ebook. In lecture, students answer Peer Instruction questions in the ebook, solve Parsons problems, and work in groups to solve POGIL-style activities.

**Interactive ebooks** for computing courses typically allow students to run code and display the result in the ebook. They include support for typical instructional content such as videos, text, and images. Many also

IMAGE BY MASTER1305

support a wide range of additional practice problems such as multiple-choice questions, fill in the blank questions, matching questions, code writing problems with unit tests, and mixed-up code (Parsons) problems. These problems provide immediate feedback and can be automatically graded. There are both free ebooks (for example, from Runestone Academy and OpenDSA) and commercial ebooks (for example, from Zybooks and Codio). Interactive ebooks improve learning versus static textbooks, and students prefer them. I have been creating free interactive ebooks on the Runestone Academy platform for many years. I use one of these ebooks, *Python for Everyone—Interactive*, in my course. A static version of this ebook was originally created by Charles Severance for his very popular *Python for Everybody* online course. Undergraduate students and I modified his static ebook to make the code examples runnable and to add lots of interactive practice problems as well as new content.

**Peer Instruction (PI)** was originally developed by Eric Mazur of Harvard University to improve students' understanding in physics. Sometimes, the name "Peer Instruction" leads to confusion with peer instruction (working with peers). In PI as defined by Mazur, students read material before lecture and do an assignment or a quiz based on the reading either before lecture or at the beginning of lecture. During lecture the instructor displays a hard multiple-choice question with distractors (incorrect answers) based on common student misconceptions. Students answer the question individually (first vote), then discuss their answer with peers, and then answer individually again (second vote). Next the instructor shows the outcome of the two votes and leads a discussion about the question. There are many variants of Peer Instruction. Instructors do not always assign readings before lecture or include an assessment of the reading. Instructors may skip the second vote, especially if the percentage of students who got the question right on the first vote is above 70% or below 30%. Learning is optimized when about half of the students get

**Active learning is particularly effective for students from historically marginalized groups.**

**Solving Parsons problems is typically significantly faster for students than writing the equivalent code and results in similar learning gains.**

the question wrong on the first vote. There are a variety of ways to vote including raising hands, using cards, or using electronic devices such as iClickers (handheld devices that look like television remotes).

I was introduced to Peer Instruction (PI) at computing education conferences where I learned of approximately a decade of positive research results for PI in many STEM subjects including computing.[1] I also attended a workshop that explained the research, demonstrated the process, and provided resources. One of the resources was a website (see http://peerinstruction4cs.com/) with slides for several computing courses with embedded PI questions. When I first tried PI, I had students vote with iClicker devices. However, after every lecture I would have a line of students who had forgotten their iClicker devices or the batteries had run out, but still wanted credit for answering the Peer Instruction questions. Later there was an app that students could use to answer PI questions on their phones or laptops, but the app wasn't free. Since I was already creating and using an interactive ebook for the course, I decided to create a new tool to support answering PI questions by building it into the Runestone platform.

This tool supports both in-person discussion and remote discussion via a chat interface. The chat interface allows us to maximize the number of groups that have members with different answers. It is also useful when a lecture is held on Zoom. The tool also includes an interface for students who answer PI questions outside of lecture. That student votes individually and then enters a justification for their answer. They then view a saved chat discussion where one of the members answered the same way they did, and then they vote again (still individually). We are currently studying the effectiveness of this new tool.

I first heard about mixed-up code (Parsons) problems at the 2012 ICER conference just after I had started my Ph.D. In a Parsons problem, students place mixed-up blocks in order to solve a problem see the accompanying figure). Parsons problems can also have distractors blocks that are not needed in a correct solution. I expected Parsons problems would be an easier form of practice than writing code from scratch. I added Parsons problems to a free ebook on the Runestone platform and watched what happened. I found that more students attempted to answer the Parsons problems than nearby multiple-choice questions. This encouraged me to study Parsons problems in my dissertation work. However, I also found that some students really struggled to answer Parsons problems. Some gave up and never answered them. Practice must be successful for learning to occur, so I wanted to guide students to the correct solution without just giving them the solution. I added a form of adaptation trigged by clicking a "Help Me" button after at least three incorrect attempts. Each time the student clicks the "Help Me" button it will either remove and disable a distractor block, or if there are no distractor blocks in the solution area, it will combine two blocks into one until the solution area has only three blocks.

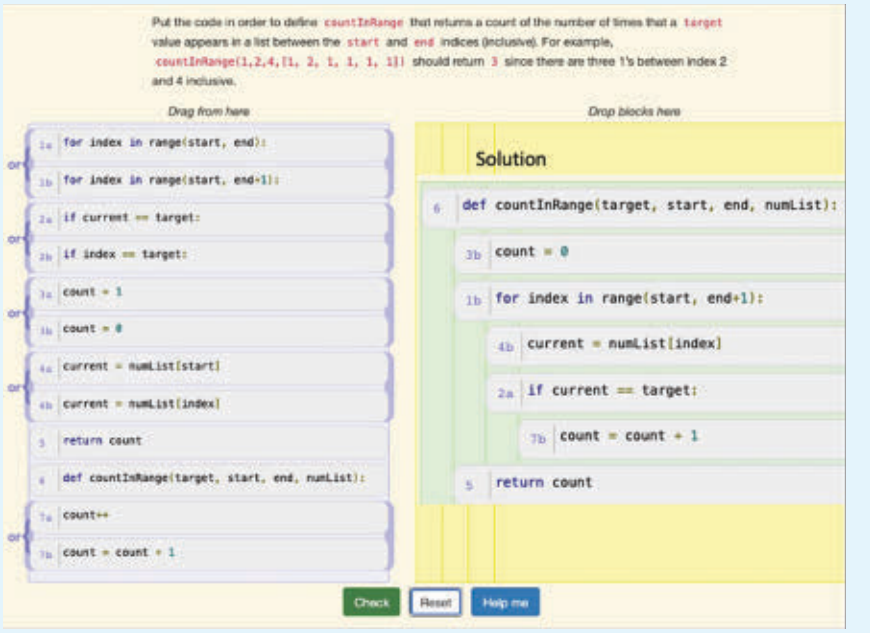
Research on Parsons problems has been growing.[2] Several online systems support Parsons problems including Runestone Academy (see https://runestone.academy/ns/books/index), PraireLearn (see https://www.prairielearn.org/), Epplets (see https://epplets.org/), and Codio (see https://www.codio.com/). Studies have provided evidence that most students

find solving adaptive Parsons problems useful for learning to program and perceive them as easier than writing the equivalent code. Solving Parsons problems is typically significantly faster for students than writing the equivalent code and results in similar learning gains. There is evidence that the distractors help novices learn to recognize common syntax and semantic errors. A log file analysis of ebooks with both adaptive and non-adaptive Parsons problems found that learners where nearly twice as likely to correctly solve an adaptive Parsons problem than a non-adaptive one. However, some students would rather write code from scratch than solve a Parsons problem, especially students with more prior programming experience. We added a new type of toggle problem to Runestone Academy that allows students to choose to solve either a Parsons problem or the equivalent code-writing problem. We also created another type of toggle problem that allows students to pop-up an adaptive Parsons problem when they are struggling to write code. Students can solve the Parsons problem but they still must at least type the solution in the code writing problem area. We are currently studying both of these types of toggle problems.

One year at SIGCSE I attended a Process Oriented Guided Inquiry Learning (POGIL) workshop. POGIL is a form of group work that is carefully structured to allow students to actively discover important concepts. Research on POGIL has provided evidence that it improves teamwork skills and learning.[4] There is a CS-POGIL website (see https://cspogil.org/Home) with worksheets from several researchers and for several courses and languages. Students work through the worksheets in structured groups. I integrated POGIL-style activities into the ebook that I use in my course (*Python for Everybody—Interactive*) so that the questions provide immediate feedback and can be automatically graded. Over time, I changed most of my lectures into POGIL-style activities. When students work on a POGIL-style activity I ask them to record what they learned and any questions in a Padlet. After they have worked in groups, I go over the questions. I have seen a big improvement in student engagement in lecture when students work on these activities versus passive lecture.

**My goal is to help more students succeed in computing courses, especially those from groups that do not have access to computing courses before college.**

**A Parsons problem with the mixed-up blocks is on the left. Notice there are pairs of correct and distractor code with purple edges grouping them and an "or" to indicate the students should pick. The correct solution is shown on the right.**




Active-learning techniques including interactive ebooks, Peer Instruction, mixed-up code (Parsons) problems, and POGIL can be used to improve student learning, skills, and motivation in computing courses. These approaches work well even large courses with hundreds of students. This is important since the number of students in introductory computing courses has more than tripled at many institutions since 2006. My goal is to help more students succeed in computing courses, especially those from groups that do not have access to computing courses before college. I encourage computing instructors to try one or more of these approaches. I find that these approaches are not only better for students, but also increase my enjoyment of lecture since students are more engaged.


**References**

1. Crouch, C.H. and Mazur, E. Peer instruction: Ten years of experience and results. *American Journal of Physics 69*, 9 (Sept. 2001), 970–977.
2. Ericson, B.J. et al. Parsons problems and beyond: Systematic literature review and empirical study designs. In *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol. 2* (Dublin, Ireland, 2022). ACM, New York, NY, USA.
3. Freeman, S. et al. Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences, 111*(23 (2014), 8410–8415.
4. Yadav, A. et al. Collaborative learning, self-efficacy, and student performance in cs1 pogil. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. (Mar. 2021), 775–781.


**Barbara Ericson** (barbarer@umich.edu) is an assistant professor on the School of Information at the University of Michigan in Ann Arbor, MI, USA.

DOI:10.1145/3576931

George V. Neville-Neil

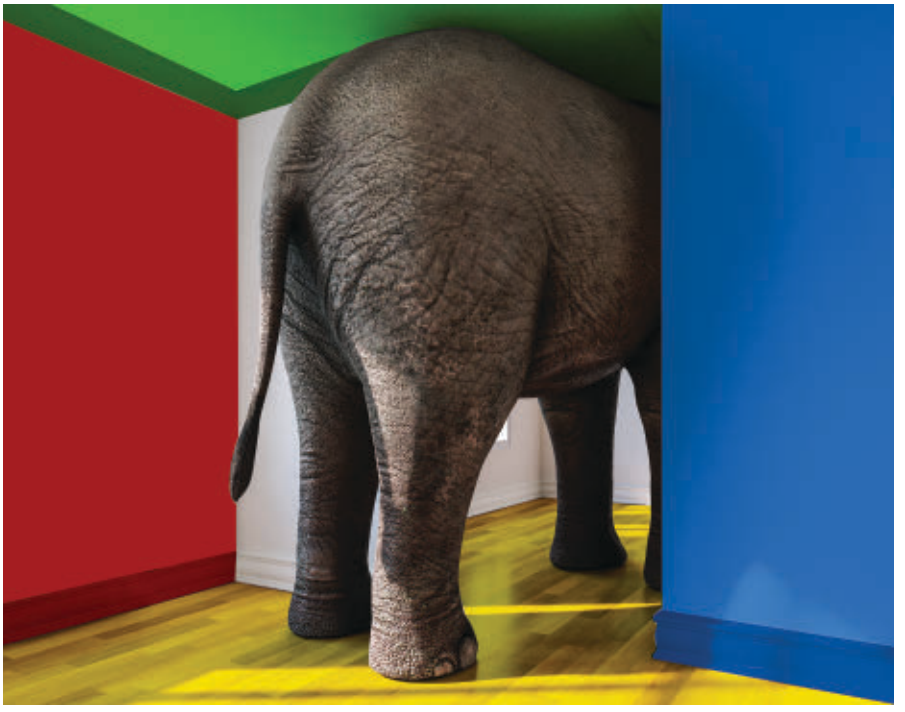
# Kode Vicious
# The Elephant in the Room

*It is time to get the POSIX elephant off our necks.*

**Dear KV,**

While working with our data science team, I began to notice something peculiar about their code, which is a combination of Python and C/C++ libraries as you find in iPython or Jupyter notebooks. The amount of code in their programs that is responsible for getting data to where they can work on it almost always overwhelms the amount of code that operates on the data itself. And this intellectual overload, which they rightly think they should not worry about, is a drag on their overall productivity. If they had a way to just operate on the data rather than fooling around with finding, opening, reading, writing, and closing files, to take one example, let alone managing their program's memory when they delve into C and C++ libraries, it seems it would suit them a lot better. When I talk to the team, they just say they accept this situation, because their two choices seem to be either using a framework that's so high-level that the performance is poor, or using a system with all the lower-level knobs exposed that—while fast—is error-prone and exposes them to a lot of the system plumbing. Surely there is a better middle ground somewhere?

**All Plumbed Out**



**Dear Plumbed,**

You ask a deep question: "Why do our programs take the shape in which we see them?" KV often waxes historical and will, alas, do so again. Software and hardware have followed a random walk over the history of computing. Hardware often leads and software often follows, although from time to time, software has forced itself upon the hardware side to get new ideas to execute at the speed of the hardware. Consider the history of Virtual Memory and Memory Management Units, or RISC processors that are meant to run that ever-popular programming language, C, as fast as possible.

The plethora of plumbing you see in most programs is due, in part, to an elephant that has been sitting on our collective backs for nearly 40 years: Posix, which is just a nice way of saying Unix, and even though they disavow it, Linux. Only two models of programming are in common use by anyone not working inside another program (did you know people think programming in Excel is programming?) and these are Posix/Unix/Linux and Windows, at least for most programmers. All these systems grew up in an era of hardware that was very different from our own. This is evidenced in the way that programmers are forced to interact with them. Let's consider two examples: getting at data and sharing data between programs or threads.

The point of a program is to transmute data between states, a fact often

IMAGE BY R.CLASSEN

lost when a programmer comes to write the code, because first, as you point out, they must deal with the plumbing. Current systems were developed in an era where secondary storage, aka spinning rust, was often provided in disk drives the size of a washing machine and was larger by several orders of magnitude than the system's main memory.

The secondary storage was also slow and unreliable. Unix, which eventually became codified as Posix, created a huge amount of plumbing around secondary storage to make it more reliable and aid the programmer in finding just what it was they had stored in the first place. Lest you think that the designers were trying to make things more complex, you are quite wrong. The systems for handling secondary storage that preceded the *NIX age were complex and not portable, so the changes were most welcome, but once Posix had established itself in the industry, it acted as a huge elephant that prevented innovation and progress.

The classic example of this is embedded systems software in which there are many and varied kernels, some of which provide truly innovative features not found in typical desktop or server systems. The death of any of these systems is when a user asks, "What about Posix? How can I port my XXX program to run on your system?" Providing Posix-like semantics in these systems is their death knell, because the Posix way of thinking is so narrow and providing its varied illusions requires so many hidden changes and adherences to age-old assumptions that there is no way to have a flexible innovative system and serve the Posix elephant. This means that no matter how innovative and clever a system is, once it is tainted by Posix support, it becomes just another Posix system. And then we have the same plumbing, provided in the same way as before, and the innovative bits of allowing the programmer new ways to transmute their data gets lost in the noise.

The second example of how software plumbing has not kept up with the times is the way programs work with shared memory. If you look at most interprocess communication mechanisms, you can see that they are most often used by only two programs—usually a client and server. The Socket API, local IPC, and shared memory pretty much assume two programs, and the addition of threads to these programs is fraught with peril and often poorly provided for by programming languages. A two- (or small number-) process model of working with shared memory makes perfect sense when there is only one CPU in each computer, but the days of single-core computers went out with cellphones the size of bricks.

**Once Posix had established itself in the industry, it acted as a huge elephant that prevented innovation and progress.**

You would be hard-pressed to find a computer (and definitely not a server) in which there were not tens of CPUs. And it is not as if we were not all warned about this explosion of cores. In fact, this very magazine published about the coming of multicore systems nearly 20 years ago, and yet our way of providing programming APIs for this new world is based on APIs developed when computers rarely had more than one CPU. The pthreads API (part of Posix) remains the most common way for programmers to write code that can take advantage of current multicore designs—an API so difficult to use papers have been written about how no programmer should ever try to use it.

Most attempts to handle the plumbing problem have followed the age-old software paradigm of adding yet another abstraction. Hiding the plumbing is good, but once your toilet overflows for the 10th time, maybe it's time to change the plumbing rather than buy a longer snake.

What is required to get away from this plumbing problem is to rethink how programs and data interact in modern systems; a topic that Poul-Henning Kamp, a frequent contributor to *Communications*, covered in a talk more than a decade ago about how we should stop programming computers as if they were PDP-11s. If we look at a modern computer, it has several features that are not covered in software design classes, most of which still assume the single-core, small-RAM, big-disk model. Now, all computers have many cores. Most computers have main memories that dwarf the secondary storage of the systems used when your OS textbooks were written. The secondary storage is largely reliable and does not depend on flying heads. In such a world, we should be able to think about programming for our data rather than programming for our plumbing.

We should be able to assume that the data we want exists in main memory without having to keep telling the system to load more of it. APIs for managing threads and access to shared memory should be rethought with defaults created for manycore systems, and new schedulers have to be built to handle the fact that memory is not all one thing. Modern systems have fast caches near CPU, main memory, and flash memory. Soon we will have even more memory, disaggregated, which is faster than disk but slower than main RAM.

If we are to write programs for such machines, it is imperative to get the Posix elephant off our necks and create systems that express in software the richness of modern hardware. Only then will programs be about data, which they should be, and not about the plumbing to get at the data.

**KV**

**Related articles on queue.acm.org**

**Cloud Calipers**
*Kode Vicious*
https://queue.acm.org/detail.cfm?id=2993454

**Crash Consistency**
*Thanumalayan Sankaranarayana Pillai, et al.*
https://queue.acm.org/detail.cfm?id=2801719

**The Most Expensive One-Byte Mistake**
*Poul-Henning Kamp*
https://queue.acm.org/detail.cfm?id=2010365

**George V. Neville-Neil** (kv@acm.org) is the proprietor of Neville-Neil Consulting 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.

DOI:10.1145/3576932

Deborah G. Johnson and Mario Verdicchio

▸ Susan J. Winter, Column Editor

# Computing Ethics
# Ethical AI Is Not about AI

*The equation Ethics + AI = Ethical AI is questionable.*

IMAGE BY FRAN_KIE/SHUTTERSTOCK

Many scholars and educators argue the antidote to some of the ethical problems with artificial intelligence (AI) is to integrate ethics and AI or embed ethics in AI.[2,12,14] The product of this combining is supposed to lead to Ethical AI, a term that is both frequently used and seemingly elusive.[5,9,13] Although attempts to make AI ethical are to be lauded, too little attention has been given to what it means to "integrate" or "embed," be it integrating ethics and AI or embedding ethics in AI.

A rather simple idea of additivity seems to be behind these proposals. That is, the efforts are directed toward figuring out how ethical principles can be "injected into"[11] AI or how an ethical dimension can be "added to" machines[1] or, if the focus is on the latest wave of machine learning, how to "teach" machines to act in an ethical way.[10] The common vision of such proposals is ethical components can and should be added to existing AI systems. Representing this vision with an equation gives us Ethics + AI = Ethical AI.

However, the truth of this equation is questionable. Additivity between two entities requires ontological likeness. Adding ethics and AI is based on ontological assumptions about what AI is and what ethics is, namely that the two entities are of the same nature or, at least, some of their components are.

As a discipline, AI was born as a subfield of computer science, so naturally AI artifacts, such as algorithms (for example, resolution for automated reasoning), models (for example, artificial neural networks for machine learning), software (for example, the Eliza chatbot) or hardware (for example, neuromorphic chips) are all computational in nature. Among the founders of the discipline, we may even find computational stances regarding human intelligence.[8] Questions on the nature of human intelligence aside, all the artifacts that ethics is supposed to be "injected into" or "added to" or "taught to" are software running on computers, that is, bona fide computational entities. If the nature of AI in the Ethics + AI addition is computational, then this seems to entail that ethics or, at least, the ethics that we add to AI must be computational as well. In other words, if AI is a piece of software, then for that software to become ethical it will have

to include instructions that express ethics in computational language.

Whether ethics is, at its core, computational is a deep question and there are good reasons to be skeptical. Those who are now trying to code ethics tend to think of it as rules or principles or theories. This way of thinking is attractive because rules, principles and theories are amenable to formalization, and formalized systems can be translated into computational ones, some theoretical boundaries on the completeness of such translation notwithstanding.[4] However, people (individuals, societies, cultures) do not generally act on the basis of adherence to philosophical moral theories, and although individuals may acknowledge and embrace moral rules and principles (for example, respect human life, treat others fairly), rules and principles must always be interpreted and applied.

Ethics eludes computation because ethics is social. The concepts at the heart of ethics are not fixed or determinate in their precise meaning. To be applied they must be interpreted, and interpretations vary among individuals and groups, from context to context, and may change over time. Yes, we all agree that respect for persons, fairness, and keeping promises are good or even essential to ethical behavior, but in real-world situations and particular domains, these concepts require interpretative specification. For example, in medicine, analysis is required to figure out what respect for persons could mean for doctor-patient relationships. It was not until the 1970s that respect for persons was translated into informed consent and doctors later were prohibited from experimenting on patients without it. Privacy is an even more complex concept with a good deal of controversy about how it can or should be protected in practice. Many companies have interpreted it to mean they must have privacy policies containing elaborate details that customers never read but agree to by default; others argue that this interpretation is not an adequate understanding of privacy. Sometimes social consensus on an interpretation leads to a law, for example, equality and justice mean (among other things) non-discrimination; other times social consensus leads to informal social conventions, for example, keeping one's promises is implicit in many interpersonal relationships; and yet other times, the interpretation of a social value continues to be contested and unsettled, for example, how is universal suffrage to be achieved.

**Even though ethical concepts are not amenable to computational expression, there are often computable dimensions to ethical problems.**

Because ethical concepts require social interpretations, they are subject to disagreement, contestation, and change. No computational model can capture all the possible interpretations that can constitute the social meaning of an ethical concept. As such, ethical concepts are not conducive to computational expression. Some may argue that a computational model could count as a particular interpretation of an ethical concept, but even that would only be the case if there were social understanding and acceptance of the meaning of that interpretation.

Nevertheless, even though ethical concepts are not amenable to computational expression, there are often computable dimensions to ethical problems. Computability is the characteristic of problems that can be described in a mathematical form that is compatible with the operations of a computer. Computable problems can be given to a computer as input, and computation can provide solutions to them. Such solutions can solve part of an ethical problem, but not all of it, since the problem will have aspects that elude computation. In other words, the role of computation in solving ethical problems will be limited in scope; it will not be able to incorporate ethical notions which depend on variable social agreement or ethical ideas that have been interpreted in diverse and contested ways or may be in flux.

**We can continue to think about AI as merely computational artifacts, but we should acknowledge there is another, more complex notion of AI.**

Consider the following example. Accusations of bias in algorithmic systems are usually based on ideas about equity, fairness, and non-discrimination. One context in which this arises is in recruitment and hiring and within this context, one area that is often thought to be solvable by algorithms is in the distribution of job advertisements.[7] The basic idea is that an equal distribution of ads among all demographic categories ensures a fairer job market for example, one that does not discriminate. However, although an algorithm can produce a distribution of data (ads) that represents an interpretation of equality, this does not necessarily make the hiring system fair. For one thing, whether or what kind of equity is achieved depends on the demographic categories and venues used in the computational distribution. For another, the distribution of ads is only one part of fairness of job markets; the rest requires non-computational strategies, be they organizational, governance, design and use practices, social change or all of these factors.

This means ethics cannot be added to AI—if AI is understood to be computational artifacts—because ethics cannot be understood as purely computational. However, there is another way to conceptualize AI. AI artifacts are generally used in contexts in which they are part of social practices that involve human behavior, goals, and norms. An important dimension of this is that the

output of an algorithm computed by a machine has significance and efficacy only insofar as human beings attach meaning to it and use it.

Borrowing a concept that is a staple of Science, Technology and Society (STS) studies, technologies, including AI are more productively understood as sociotechnical systems, that is, systems in which artifactual behavior is combined with human, social, and organizational behavior[3] to produce results. Viewing AI as sociotechnical systems provides a broader scope to our understanding of AI. It does not deny or ignore the fundamental and defining contribution of computation to AI, but it takes into consideration the important relations that hold between AI artifacts and the people who design them, those who deploy them, those who make policies about them and, ultimately, those who use them. Neglecting these relationships is an oversight that we named sociotechnical blindness.[6]

Of course, we can continue to think about AI as merely computational artifacts, but we should acknowledge there is another, more complex notion of AI. With the sociotechnical systems understanding of AI, AI artifacts are understood to be components in systems that are constituted by social practices, social norms, and social meanings as well as the computational artifacts. With the sociotechnical systems concept of AI, AI has ethical significance. Returning to our job recruiting example, the algorithm may distribute ads equally but whether "equally" constitutes fairness or nondiscrimination depends, as mentioned earlier, on the nature of the categories as well as such factors as the content of the ad, how many likely qualified individuals have internet access, how people and disciplines think about discrimination, and so on and on and on.

The two concepts of AI—one narrow and only referring to the computational artifacts and the other broader and including the social arrangements in which AI artifacts operate—should not be conflated; the lens of ethics can only be turned to AI understood as sociotechnical systems. An AI artifact cannot be ethical or unethical, good or bad, biased or unbiased. With the broader concept of AI, AI artifacts can be understood as having an ethical dimension, not per se, but as part of a sociotechnical system.

**The challenge for AI experts is to acknowledge that the organizations and industries in which algorithms operate are far from morally perfect.**

Many of the current notions of Ethical AI miss the mark on this, not thinking about AI as sociotechnical and not acknowledging that AI has ethical dimension only insofar as it affects social relationships and arrangements and impedes or furthers social values. Ethics cannot be added to AI artifacts but such artifacts can be components in systems that have ethical significance, that is, systems that can be evaluated in ethical terms. Does the job recruiting system fairly distribute job advertisements? Does the parole system that determines which convicts are eligible for parole do so without discrimination? Do social services decide the eligibility of applicants accurately? The algorithms that these organizations use in making their decisions are part of the ethical question but they are only part of and only ethical in conjunction with other practices in the system.

The crux of the matter is that we cannot have ethical AI unless we have ethical domains or industries in which AI algorithms operate. If the norms by which a company or industry operates are unfair then the AI artifacts that instrument some of those activities will be unfair, and we will not be able to right that wrong only by means of computation.

The challenge for AI experts is to acknowledge that the organizations and industries in which algorithms operate are far from morally perfect. Indeed, there are no morally perfect systems (though some are better than others with respect to specific criteria). Seeking a better, fairer, more just and more humane world is a project, a project which AI experts can (and many already do) embrace, and a project for which they have a great deal to contribute. However, this generally involves *more* than working with machines, albeit intelligent machines. AI experts must pay attention to, and critically examine, the operations of the enterprises for which they are designing AI and try to improve on them. The interpretive fluidity and potential contestation of the ethical concepts make this especially challenging. Nevertheless, despite all difficulties and no promise of success, striving for Ethical AI is the right thing to do. C


**References**

1. Anderson, M. and Anderson, S.L. Machine ethics: Creating an ethical intelligent agent. *AI Magazine 28*, 4 (Apr. 2017), 15.
2. Arkin, R.C. Governing lethal behavior: Embedding ethics in a hybrid deliberative/reactive robot architecture. In *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction* (2008), 121–128.
3. Baxter, G. and Sommerville, I. Socio-technical systems: From design methods to systems engineering. *Interacting with Computers 23*, 1 (Jan. 2011), 4–17.
4. Gaifman, H What Gödel's incompleteness result does and does not show. *The Journal of Philosophy 97*, 8 (Aug. 2000), 462–470.
5. Gibney, E. The battle for ethical AI at the world's biggest machine-learning conference. *Nature 577*, 7791 (2020), 609–610.
6. Johnson, D.G. and Verdicchio, M. Reframing AI discourse. *Minds and Machines 27*, 4 (2017), 575–590.
7. Lambrecht, A. and Tucker, C. Algorithmic bias? An empirical study of apparent gender-based discrimination in the display of STEM career ads. *Management Science 65*, 7 (2019), 2966–2981.
8. McCarthy, J. What is artificial intelligence?" Computer Science Department, Stanford University (2007); https://stanford.io/3uVskIX
9. Mittelstadt, B. Principles alone cannot guarantee ethical AI. *Nature Machine Intelligence 1*, 11 (2019), 501–507.
10. Noothigattu, R. et al. Teaching AI agents ethical values using reinforcement learning and policy orchestration. *IBM Journal of Research and Development 63*, 4/5 (2019), 1–9.
11. Rossi, F. and Mattei, N. Building ethically bounded AI. In *Proceedings of the AAAI Conference on Artificial Intelligence 33*, 01 (2019), 9785–9789.
12. Saltz, J. et al. Integrating ethics within machine learning courses. *ACM Transactions on Computing Education (TOCE) 19*, 4 (Apr. 2019), 1–26.
13. Siau, K. and Weiyu, W. Artificial intelligence (AI) ethics: Ethics of AI and ethical AI. *Journal of Database Management (JDM) 31*, 2 (Feb. 2020), 74–87.
14. van de Poel, I. Embedding values in artificial intelligence (AI) systems. *Minds and Machines 30*, 3 (Mar. 2020), 385–409.



**Deborah G. Johnson** (dgj7p@virginia.edu) is Olsson Professor of Applied Ethics, Emeritus, in the Department of Engineering and Society at the University of Virginia in Charlottesville, VA, USA.

**Mario Verdicchio** (mario.verdicchio@unibg.it) is a researcher at the University of Bergamo, Italy, and a member of the Berlin Ethics Lab at the Technische Universität Berlin, Germany.

DOI:10.1145/3539783

Charles Isbell, Michael L. Littman, and Peter Norvig

# Viewpoint
# Software Engineering of Machine Learning Systems

*Seeking to make machine learning more dependable.*

IMAGE BY ARTEMISDIANA

Machine learning (ML) is ubiquitous, contributing to society-facing applications that each day impact how we work, play, communicate, live, and solve problems. In 2015, after Marc Andreesen proclaimed "software is eating the world," others countered "machine learning is eating software." While there have been exciting ACM A.M. Turing-award-winning worthy advances in the science of ML, it is important to remember ML is not just an academic subject: it is a technology used to build software that is consequential in the real world.

Failures of ML systems are commonplace in the news: IBM's Oncology Expert Advisor project is canceled for poor results after a $60 million investment; the chatbot Tay learns abusive language; an Uber self-driving car runs a red light; a Knightscope security robot knocks over a toddler;[8] facial recognition systems unfairly make three times more errors for non-white non-males.[2]

For 50 years, the software engineering (SE) community has been learning from its own failures and introducing new tools, techniques, and processes to make better software. To build robust, reliable, safe, fair systems with ML, we need all these established SE techniques, and some new ML-specific ones. In turn, software engineers should realize how ML can change the way software is developed. Our goal in this Viewpoint is to pull together some of the threads that relate to this topic to spur more conversation about putting these ideas into practice. Ignoring these insights will lead to unprecedented levels of technical debt. In brief, we believe that ideas from software engineering such as explicit requirements, testing, and continuous integration need to be embraced by teams that are making systems built on top of data.

## Software Engineering Basics

In some organizations, machine learning researchers develop ML algorithms using R or Python notebooks without basic software engineering tools like type checking, reusable modules, version control, unit tests, issue tracking, system and process documentation, code reviews, postmortems, automated hermetic builds, and continuous monitoring. This lack of discipline is an avoidable source of failures—organizations should insist on standard SE practices from their ML teams. ML researchers should be encouraged to use the same tools and pipelines that will be used in production, heeding the emerging machine learning operations (MLOps) techniques. As the NASA saying goes, "Test what you fly, fly what you test." Including software engineering education as part of the data science curriculum will help newly mint-

ed students appreciate how important these processes are to help them meet their responsibilities.

### Taking Data Seriously

Every software program deals with issues of verifying data, keeping it secure and private, and scaling up to big data when needed. But in ML the stakes are raised because algorithms learn from the data, so a data error can lead, not just to a problem for one customer, but to decreased performance for every customer.

Organizations should employ a data pipeline that tracks the provenance of all data, allows it to be annotated, managed, and abstracted, and protects the privacy of personally identifiable or proprietary data. Promising technologies including differential privacy, federated learning, and homomorphic encryption are key tools for privacy, but are in varying stages of maturity .

ML teams must have separate training, validation, and test datasets, which should not accidentally become intertwined as the design space is explored. Teams can fool themselves by running experiments until the results look good, and then stopping. At that point, they cannot be sure if the last good results are truly representative of future performance, or if the results will regress back to the mean. Teams must protect themselves by adhering to an experimental design and stopping rules.

Many ML algorithms include randomness, so different training runs may produce different results. More thorough testing is needed as a consequence. Throughout the development process, developers must acknowledge that the fingerprints of the human engineers are all over the final result, for better and for worse.

When example data comes from users, ML systems are vulnerable to mistakes and intentional attacks by these users. The site labelerrors.com catalogs labeling errors in popular crowdsourced image databases (such as labeling a *dingo* as a *coyote*), which corrupts systems that train on this data. Worse, adversaries who can contribute as little as 1% of the training data can poison the data, allowing them to exploit an ML system such as an email spam filter.[7] To guard against these attacks, we must verify data as well as software, and promote "adversarial thinking" throughout the educational pipeline so system designers know how to defend against malevolent actors.

### Optimizing the Right Thing

In traditional software engineering, the requirements say what should be achieved, and programmers design and develop an implementation that meets the requirements, verifying their choice of *how* satisfies *what* must be done. With ML, computation becomes a partner in this translation process: Programmers state *what* they want achieved by supplying training examples of appropriate (and inappropriate) behavior and defining an objective function that measures how far off the implementation is from the requirements. A learning algorithm then takes the training examples and objective function and "compiles" them into a program that ideally produces success on future examples. If we get the data and requirements right, the software "writes itself."

Unfortunately, the reality of ML is it is difficult to precisely and unequivocally say what you want. ML algorithms will exploit loopholes in the requirements. A Roomba that was instructed not to bump into furniture learned to drive backward, because there were no bumper sensors on the rear. Videogame-playing programs were instructed to achieve the highest score, and did so not by playing the game, but by writing their name in the list of high scorers.[4]

Expressing requirements as objective functions can surface their latent contradictions. For example, the law requires equitable treatment of protected classes in employment, housing, and other decision-making scenarios.[5] When writing requirements for a system to predict whether a defendant will commit a crime in the future, we need to translate this "equitable treatment" law into an objective function. One requirement is that predictions are equally accurate for, say, both Black and white defendants. Another requirement is that the harm done by bad predictions affects Black and white defendants equally. Unfortunately, it is not possible to achieve both these requirements simultaneously.[6] ML developers, in consultation with all stakeholders, must decide how to make trade-offs in objectives.

## Building Trust

A successful software system must not only get the right answers, but also convince the developers and users to trust the answers. As ML software is increasingly being tasked with highly sensitive decision making in areas such as hiring, medical treatment, lending, and criminal sentencing, we have to ensure such systems are fair, transparent, accountable, auditable, explainable, and interpretable. We must determine what sort of explanations will justify treatment of an individual; what kind of tests will certify robust system performance across likely use cases; what type of auditing and reporting will demonstrate fairness across diverse protected classes; and what remediation processes are available when there are errors. Doing so is difficult because ML systems are often used for the most complex and ill-defined tasks—if the tasks were easy, we would not need an ML solution.

For example, despite enthusiasm for ML classifiers in medical imaging and despite many systems achieving 95%+ accuracy,[1] there are still few deployed systems, because ML engineers have not yet convinced doctors the systems are robust across different X-ray machines, different hospital settings, and so forth. Without better assurance, there is insufficient reason to believe such high accuracy will persist in the wild, resulting in poor outcomes.

Traditionally, programmers try to write code that is general enough to work even in cases that were not tested. In ML systems, it is hard to predict when a training example will generalize to other similar inputs, and when the example will be memorized by the system with no generalization. We need better tools for understanding how ML systems generalize.

In traditional software, major updates are infrequent, leaving sufficient time to thoroughly test them. Small patches are localized, and thus only require testing the parts that are changed. But in ML systems such as Internet search and product recommendations, new data is arriving continuously and is processed automatically, often resulting in unbounded nonlocal changes to ML models. ML developers need to be prudent about how often they retrain their models

**We need better tools for understanding how ML systems generalize.**

on new data—too frequently and it will be difficult for quality assurance teams to do thorough testing; too rarely and the models become stale and inaccurate. We need better tools for tracking model evolution. In addition to tracking new data, we need a process for discarding old data, as when a major event like a new social media platform or a pandemic significantly alters the data models rely on. Avoiding downstream trouble due to such data cascades takes vigilance , as well as inputs from domain experts and from "softer" fields such as human-computer interaction.

Verification of traditional software relies on abstractions that let us divide the system into components that are composable, modular, interpretable, and often reusable. Current methods for building ML systems do not typically lead to models with these properties. Rather, each new ML system is built from scratch, and each resulting model is largely a black box requiring significant work to understand and verify. One promising approach is a class of multitask unified models[6] that share internal structure across related tasks, data sets, and languages. More effort can go into verifying a single large model, which can then be specialized for new tasks.

## Diverse Viewpoints

It is easy to create, say, a calculator program, because all users agree that the answer to 12×34 should be 408. But for many ML problems the desired answers are personalized for each user. That means that we need to do balanced data collection, particularly for underrepresented groups, to assure that each group gets the answers they need. We cannot let the data of the majority undermine proper answers for minority classes.

To properly serve a diverse population of users and use cases, ML development teams should be diverse themselves, and should be trained to consider diverse needs, and not to be satisfied with existing data sets that are unbalanced in their coverage.

## Society Needs Us

There can be many sources of bugs in ML programs: traditional bugs in the implementation of the algorithm; invalid, incomplete, or unrepresentative data; incompleteness in the specification of objectives; poor choices of hyperparameters; and more.

Now that ML software is rapidly expanding in its prevalence, consequence, and influence on our lives, it is clear we need careful, *professional* methodologies designed to encourage the positive while moderating the negative impacts of this technology on society. We must build tools to support these methodologies, train people to use them, and develop certification systems that reassure users and minimize failures.

Making ML dependable will require new research and new practice. It will require machine learning researchers thinking like software engineers and software engineering researchers thinking like machine learning practitioners. The consequences are too dire to do anything else. C


**References**

1. Aggarwal, R. et al. Diagnostic accuracy of deep learning in medical imaging: A systematic review and meta-analysis. *Digit. Med* . *4*, 65 (2021).
2. Grother, P. et al. Face Recognition Vendor Test (FRVT). National Institute of Standards and Technology, (Dec. 2019).
3. Kleinberg, M. et al. Inherent Trade-Offs in the Fair Determination of Risk Scores. CoRR, (2016).
4. Krakovna, V. and Uesato, J. et al. Specification gaming: The flip side of AI ingenuity. DeepMind Blog, (Apr. 2020).
5. Mehrabi, N. and Morstatter, F. et al. A Survey on Bias and Fairness in Machine Learning, CoRR , (2019).
6. Nayak, P. MUM: A new AI milestone for understanding information. The Keyword (Google Blog), (May 2021).
7. Nelson, B. et al. Exploiting Machine Learning to Subvert Your Spam Filter. In *Proceedings of LEET'08*, (2008).
8. Partnership on AI, Artificial Intelligence Incident Database (2021); https://incidentdatabase.ai/



**Charles Isbell** (isbell@cc.gatech.edu) is Dean of the College of Computing and The John P. Imlay Jr. Chair at Georgia Tech in Atlanta, GA, USA.

**Michael L. Littman** (mlittman@cs.brown.edu) is a University Professor at Brown University in Providence, RI, USA, and was on leave at Georgia Tech at the time this Viewpoint was written.

**Peter Norvig** (peter@norvig.com) is a Distinguished Education Fellow at Stanford University in Stanford, CA, USA, and Research Director at Google.

DOI:10.1145/3545111

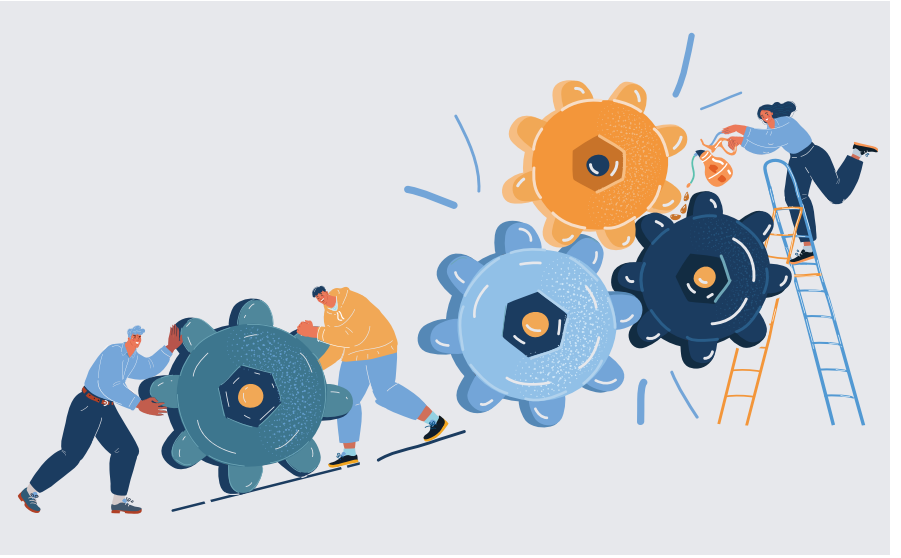
Colin Raffel

# Viewpoint
# Building Machine Learning Models like Open Source Software

*Proposing a community-based system for model development.*

TRANSFER LEARNING—USING A machine learning (ML) model that has been pretrained as a starting point for training on a different, but related task—has proven itself as an effective way to make models converge faster to a better solution with less-labeled data. These benefits have led pretrained models to see a staggering amount of reuse; for example, the pretrained BERT model has been downloaded tens of millions of times.

Taking a step back, however, reveals a major issue with the development of pretrained models: They are never updated! Instead, after being released, they are typically used as-is until a better pretrained model comes along. There are many reasons to update a pretrained model—for example, to improve its performance, address problematic behavior and biases, or make it applicable to new problems—but there is currently no effective approach for updating models. Furthermore, since pretraining can be computationally expensive (for example, training the GPT-3 model was estimated to cost millions of dollars), many of the most popular pretrained models were released by small resource-rich teams. The majority of the ML research community is therefore excluded from the design and creation of these shared resources. Past community efforts such as the SahajBERT and Big Science BLOOM models show there is a desire for community-led model development, but there is no mature tooling to support their development or continual improvement. Contrast this with the development of open source software. For example, if the open source Python programming language had remained frozen after its first release, it would not support Boolean variables, sets, Unicode, context managers, generators, keyword arguments, or many other widely used features. It also would not include thousands of bugfixes that were contributed by members of the distributed community of Python developers over the past few decades. This kind of large-scale distributed collaboration is made possible through a mature set of tools and concepts, including version control, continuous integration, merging, and more.



This Viewpoint advocates for tools and research advances that will allow pretrained models to be built in the same way that we build open source software. Specifically, models should be developed by a large community of stakeholders that continually updates and improves them. Realizing this goal will require porting many ideas from open source software development to the building and training of pretrained models, which motivates many new research problems and connections to existing fields.

IMAGE FROM SHUTTERSTOCK.COM

### Incremental and Cheaply Communicable Updates

Modern machine learning models are often trained through gradient descent. Crucially, gradient descent typically involves updating all of the parameters at every training iteration. As such, communicating any change made during training requires communicating every parameter's value. Unlike the incremental patches used to update code in version control, gradient descent makes it infeasible to keep track of the complete history of every parameter value. Fortunately, a variety of past work has shown it is possible to effectively train models using small and cheaply communicable parameter updates. These techniques could provide a helpful starting point for enabling community-developed models.

One motivation for communication-efficient training has been to reduce costs in distributed optimization, where individual workers train local copies of a model and must communicate their changes to a centralized server. This has led to schemes that either reduce the frequency of communication (that is, having workers train for more iterations before they communicate their updates) or reduce the size of each update through quantization, sparsification, or other approximations.[4,10]

It has also been demonstrated that a model can be improved or changed by adding or removing parameters. One example is the Net2Net framework of Chen et al.,[2] which describes ways of adding parameters to a neural network that does not change the function the model originally computed. Relatedly, adapters[7] are tiny trainable subnetworks that are added to a model. By only updating the adapter module parameters and leaving the rest of the model's parameters fixed, the model can be trained in a parameter-efficient manner.

Finally, certain "modular" model architectures may be more amenable to cheaply communicable updates because their architecture allows selectively updating a subset of parameters. For example, models that involve conditional computation[1] among submodules (such as the sparsely gated mixture-of-experts layer[9]) can have individual submodels added, removed, or updated while the remainder of the parameters remain fixed. Alternatively, models that make predictions by querying a collection of data (such as a *k*-nearest-neighbor classifier) can be incrementally modified by changing the data itself.

**Models should be developed by a large community of stakeholders that continually updates and improves them.**

### Merging Models

In distributed open source software development, "merge conflicts" occur when a contributor changes an out-of-date copy of the codebase or when multiple contributors introduce conflicting changes. A similar situation arises in distributed optimization of machine learning models: Since individual workers compute many local updates before communicating their changes, the individual models on each worker can become significantly different over time. A strong baseline for combining (or "merging") disparate updates from different workers is to average together the updates from each of the workers when aggregating changes,[5,6] though averaging can degrade performance when individual workers are training on differently distributed data. Using specialized model architectures could also provide a more principled way of combining updates. For example, updates to different submodels in a model using conditional computation would not conflict. Further investigation into modular model architectures will help clarify the situations where updates can be merged without a degradation in performance.

### Testing Proposed Changes

Given the ability for contributors to propose changes to a model, a natural question arises as to how to decide when maintainers should accept a change. In open source software development, this is typically done through the aid of automated testing, where a proposed change undergoes a battery of public tests to make sure the software would continue to work as intended. In contrast, the utility and validity of a machine learning model is typically measured in terms of whether it suits a given application and delivers satisfactory results, which is less straightforward to test.

A possible means of testing a pretrained model would be to measure its performance after fine-tuning on a downstream task. However, fine-tuning a model on a task typically requires many iterations of training and the utility of a pretrained model is typically measured on dozens of tasks. Testing each change to a community-developed model through fine-tuning and evaluation on downstream tasks could therefore become excessively expensive. One way to mitigate this cost would be to make fine-tuning dramatically cheaper. For example, if we fine-tune a pretrained model, perform additional pretraining, and then fine-tune it again, we might hope to reuse the updates from the first fine-tuning run. This idea bears some similarity to "merging models" in the sense that we want to merge updates from the first fine-tuning run into the updated pretrained model. We also might hope to use some of the advantages of modular architectures—for example, adapter modules created for a given pretrained model might be reusable after the model undergoes additional pretraining. Alternatively, multitask learning (where a model is trained to be able to perform many tasks) provides an alternative to the pretrain-then-fine-tune paradigm, which could make testing as simple as evaluating the model on all of the tasks it targets. Recent results have shown pretrained language models can perform well on many tasks at once,[8] though multitask performance often lags behind the performance of models trained separately on each individual task.

Even with cheaper fine-tuning and evaluation, determining how to rigorously test a contribution will require clarifying what it means to test machine learning models. As Christian Kästner noted,[3] "the challenge is how to evaluate whether the model is 'acceptable' in some form, not whether it is 'correct'."

Developing ways to specify the intended uses and limitations of models will help maintainers articulate what kinds of contributions are welcome.

### Versioning, Backward Compatibility, and Modularity

The semantic versioning system provides a standardized way of communicating the significance of a new version of a piece of software. A similar system could likely be devised for community-developed models. For example, patch releases could correspond to changes that only modify (a small subset) of a model's parameters while guaranteeing no significant degradation of performance on supported downstream tasks, minor releases could possibly change the architecture of the model while retaining the majority of its parameters, and major releases could completely replace the existing model and its parameters while targeting the same use-cases. "Backward compatibility" of a pretrained model most naturally maps to the notion of maintaining intended performance on targeted downstream tasks and that the model is applicable to the same types of inputs and outputs.

The ability to incorporate open source packages into a piece of software allows developers to easily add new functionality. This kind of modular reuse of subcomponents is currently rare in machine learning models. In a future where continuously improved and backward-compatible models are commonplace, it might be possible to greatly improve the modularity of machine learning models. For example, a core "natural language understanding" model could be augmented with an input module that allows it to process a text in a new language, a "retrieval" module that allows it to look up information in Wikipedia, or a "generation" output module that allows it to conditionally generate text. Including a shared library in a software project is made significantly easier by package managers, which allow a piece of software to specify which libraries it relies on. Modularized machine learning models could also benefit from a system for specifying that a model relies on specific subcomponents. If a well-defined semantic versioning system is carefully followed, models could further specify which version of their dependencies they are compatible with.

### Conclusion

The use of pretrained models for transfer learning has ushered in a golden era in many applications of machine learning. However, the development of these models is still in the dark ages compared to best practices in software development. Well-established concepts from open source software development provide inspiration for methods that allow building continually improved and collaboratively developed pretrained models. These connections motivate research related to existing topics like continual learning, multitask learning, distributed optimization, federated learning, modular architectures, and more. Building on advances in these fields will provide a jump-start toward this new mode of model development. In the longer term, the ability for a distributed community of researchers to build pretrained models will help democratize machine learning by shifting power away from large corporate entities working in isolation. C

**References**

1. Bengio, Y. et al. Estimating or propagating gradients through stochastic neurons for conditional computation." *arXiv preprint arXiv:1308.3432* (2013).
2. Chen, T. et al. Net2net: Accelerating learning via knowledge transfer. *International Conference on Learning Representations* (2016).
3. Kästner, C. Machine learning is requirements engineering—On the role of bugs, verification, and validation in machine learning. *Analytics Vidhya* (2022).
4. Konecný, J. et al. Federated learning: Strategies for improving communication efficiency. *Private Multi-Party Machine Learning* (2016).
5. Matena, M. and Raffel, C. Merging models with Fisher-weighted averaging. *arXiv preprint arXiv:2111.09832* (2021).
6. McMahan, B. et al. Communication-efficient learning of deep networks from decentralized data. *Artificial Intelligence and Statistics* (2017).
7. Sung, Y.-L. et al. Training neural networks with fixed sparse masks. *Advances in Neural Information Processing Systems* (2021).
8. Rebuffi, S.-A. et al. Learning multiple visual domains with residual adapters. *Advances in Neural Information Processing Systems* (2017).
9. Sanh, V. et al. Multitask prompted training enables zero-shot task generalization." In *Proceedings of the International Conference on Learning Representations* (2022).
10. Shazeer, N. et al. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *Proceedings of the International Conference on Learning Representations* (2017).

**Colin Raffel** (craffel@gmail.com) is an assistant professor in the Department of Computer Science at the University of North Carolina, Chapel Hill Hill, NC, USA.

Readers interested in discussing and contributing to this research direction are encouraged to join our community at https://bit.ly/cccml-community

DOI:10.1145/3555367

Daniel M. Yellin

# Viewpoint
# The Premature Obituary of Programming

*Why deep learning will not replace programming.*

PHOTO BY YANISHEVSKA

DEEP LEARNING (DL) has arrived, not only for natural language, speech, and image processing but also for coding, which I refer to as deep programming (DP). DP is used to detect similar programs, find relevant code, translate programs from one language to another, discover software defects, and to synthesize programs from a natural language description. The advent of large transformer language models[10] is now being applied to programs with encouraging results. Just like DL is enabled by the enormous amount of textual and image data available on the Internet, DP is enabled by the vast amount of code available in open source repositories such as GitHub, as well as the ability to reuse libraries via modern package managers such as npm and pip. Two trailblazing transformer-based DP systems are OpenAI's Codex[8] and Deepmind's AlphaCode.[18] The former is used in the Github Copilot project[14] and integrates with development environments to automatically suggest code to developers. The latter generates code to solve problems presented at coding competitions. Both achieve amazing results. Multiple efforts are under way to establish code repositories for benchmarking DP, such as CodeXGLUE[19] and CodeNET.[20]

The advent of DP systems has led to a few sensational headlines declaring that in the not-too-distant future coding will be done by computers, not humans.[1] As DL technologies get even better and more code is deposited into public repositories, programmers will be replaced by specification writers outlining what code they want in natural language and presto, the code appears. This Viewpoint argues that while DP will influence software engineering and programming, its effects will be more incremental than the current hype suggests. To get away from the hype, I provide a careful analysis of the problem. I also argue that for DP to broaden its influence, it needs to take a more multidisciplinary approach, incorporating techniques from software engineering, program synthesis, and symbolic reasoning, to name just a few. Note I do not argue with the premise that DL will be used to solve many problems that are solved today by traditional programming methods[16] and that software engineering will evolve to make such systems robust.[17] In this Viewpoint, I am addressing the orthogonal question of using DL to synthesize programs themselves.

DP models are built by training on millions of lines of code. This code encapsulates known programming and engineering techniques for solving problems. But software is forever evolving due to the following reasons.

**New machine and network architectures.** The most dramatic changes in programming come about due to changes in underlying hardware and communication technologies. Think of Cobol for the mainframe, Visual Basic for UI/event-based client-server programs, and Java for distributed programming. A good example of how languages emerge from architectural changes can be found in Alex Aiken's PLDI 2021 keynote address that describes the evolution of new programming paradigms with the advent of specialized and heterogeneous processors for high performance computing.[2] As new architectures evolve, new sorts of programs are needed to take advantage of these architectures, but these programs do not yet exist and therefore cannot be learned. How many quantum computing programs are being produced by DL today?

**New programming frameworks for solving problems.** Change in software paradigms also occurs with the advent of new programming frameworks. Programs written today make use of digital currency frameworks, social networks, and the Internet of Things (for example, smart home appliances). These programs did not exist 15 years ago. DL itself is new; libraries and frameworks have been created to support this new programming paradigm, such as PyTorch and TensorFlow. These frameworks require intimate domain knowledge to leverage and fine-tune them. Until new programming frameworks become widespread, with well-known and often used access patterns, it is very hard for DP to make accurate use of them.

**Changes to the real world.** There are continuously new challenges facing the planet due to physical phenomena causing disruptions in climate, the depletion of natural resources, and demographic changes, for example. Advances in science and technology also affect the way we live—consider the tremendous changes in last decade to healthcare, finance, travel, and entertainment. New solutions based upon code are constantly created to address the changing real world. Although snippets of these programs may be learned, the solution must model phenomena never seen before in other programs and manipulate newly created devices, and therefore cannot be learnt by DP methods in use today.

DP will not be able to generate programs that deal with new machine architectures, new programming frameworks, or solve new real-work problems. The APIs, patterns, and programming techniques will not exist in the code repositories DP is trained on. There are other pragmatic reasons that limit the applicability of DP.

**Massive computational power.** The models built by DP are huge, containing 12 billion,[8] 41 billion,[18] or even 137 billion[4] parameters. The AlphaCode paper states "Both sampling and training from our model required hundreds of petaFLOPS days" and the Codex paper states similar statistics. Only the largest organizations have access to such computing power. Not only building DP models is compute intensive, but also sampling, which is done each time a program is synthesized. The cost of building and running large DL models has led some researchers to declare further improvements in DL are becoming unsustainable.[23] This implies segments of the market that cannot afford this massive investment are unlikely to benefit much from DP.

**Specifying a program is not easy.** The underlying assumption that AI will replace programming is that it is much easier to specify a program via natural language than to write the program. One can argue, as Dijkstra did, that using natural language to specify a program is too imprecise for programs of any complexity.[11] Although the domain chosen by AlphaCode, programming competitions, include difficult problems, their specification is relatively simple. This contrasts with many real-world systems that are orders of magnitude more complex to specify. Furthermore, in the "messy" real world some requirements are best determined by trial and error.

**The advent of DP systems has led to a few sensational headlines declaring that in the not-too-distant future coding will be done by computers, not humans.**

Codex and AlphaCode generate many candidate programs for each specification. To eliminate most of them, they rely on the specification to include input/output pairs. These pairs are used to test the candidate programs and weed out those that do not produce the correct output for a given input. But many programs do not have simple input/output descriptions. Consider a program to "check the software running on each server, and if that software can be updated to a new version, and if the server configuration satisfies the requirements of the new version, update the server with the new version." While tests are developed for programs and one could ideally use them to weed out incorrect programs, these tests are often not available in detail at the start of the project, and the tests themselves may need to be programmed.

**Program rigidity.** As anyone who has spent hours or days debugging a program knows, a single misplaced character can wreak havoc. A program cannot be almost right, it must be exactly right. This is different than writing an essay, or even making a complex argument. A few flaws here and there do not change the meaning of the composition. When AI synthesizes a new textual or visual work from millions of examples, and even puts them together in novel ways, it does not need to adhere to a strict set of rules that govern the exact format of the composition. Programs, however, are not forgiving. That is why both Codex and AlphaCode use post-processing to weed out bad programs. And this is part of the reason why I advocate integrating software engineering techniques into DP (see "Unifying software engineering and DP").

**Programming as a social endeavor.** In complex projects, the final program is far from the initial idea sketched on paper. The free flow of ideas and knowledge sharing across teams cause component boundaries and APIs to change.

Initial algorithms are found to be a poor fit for the target environment and new algorithms must be invented.

**Stale repositories.** In studies on genetic makeup of population, there are known mechanisms that dilute the genetic variation in a population, such as the Founder Effect and Genetic Drift.[13] Similarly, if DP becomes the dominant mechanism for creating new programs, code repositories used to train DP will become stale. It will lack the variation that comes from a vast set of programmers populating these repositories, each with their own unique problem-solving techniques and idiosyncratic programming styles. As the genetic variation in repositories diminishes, it will be harder for DP to discover new patterns and idioms required to adapt to changing circumstances. It is yet to be seen if techniques such as genetic algorithms can circumvent this issue.[a]

**Some domains are more likely than others to benefit from DP.** Software is pervasive today across many realms. Not all of these domains will benefit equally from DP. In this regard it is useful to consider Model-Driven Development (MDD), which appeared in the 1980s and 1990s and continues to live on in low code/no code environments. It too promised to replace programming with automatic code generation. Although it never lived up to its hype, it has proven a successful methodology in some areas.[9] It is likely DP will follow a similar trajectory.[b]

There are three main factors that will determine the success of DP in a particular domain: significant financial incentive to develop and maintain DP models, tolerable cost of making a mistake,[7] and large code repositories to train on. The Copilot paradigm, incorporating DP into widely used integrated development environments (IDEs) is a scenario likely to mature rapidly. Its wide applicability creates a large market and DP can be trained on open source repositories. Since code creation is a collaborative effort between DP and programmers, the code will be more easily trusted, and the result easily integrates into existing software development processes. AI-infused IDEs will relieve programmers of many of the tedious aspects of programming such as finding the exact APIs and libraries to use, automatically generating test cases, and finding bugs. Its ability to populate function bodies with correct—or mostly correct—code will allow programmers to become more efficient and innovative.

**I advocate a multidisciplinary approach with the objective of limiting the amount of training data required and to improve the accuracy of the results.**

Another area ripe for DP is creating framework specific programs, such as programs built upon packages in widespread use (for example, ERP, CRM, SCM and e-commerce software), where the domain is much more narrow and available skills are limited. Based upon intimate knowledge of the data model and the framework APIs, DP will be able to synthesize programs to perform domain specific tasks. This is comparable to the successful application of MDD to Domain Specific Languages.[6] When such packages are used in large corporations, there exists financial incentives to support DP applied to these domains.

On the other hand, applications tightly integrated with company-specific systems are unlikely to benefit from DP. In this case there is limited data to train on as the software embodies corporate-specific APIs and usage patterns. Furthermore, organizations often have their own standards for building systems to facilitate reuse and integration across applications[6] and to guarantee regulatory compliance. Until it is cheap enough to train DP on relatively small code bases with highly accurate results, DP will not apply to this domain. The impressive results of AlphaCode were done with careful data curation and required much DL expertise. It is doubtful that organizations will be able to curate their code repositories as carefully, and they often lack the skills required.

Areas where the cost of making a mistake is dramatic (such as healthcare, national security, and regulated environments) will also not adopt DP for code synthesis. Nonetheless it is a ripe area for using DP to find bugs and validate the correctness of the code.

While Codex and AlphaCode are great engineering achievements, to make them useful in practice, further advances are required. I advocate a multidisciplinary approach with the objective of limiting the amount of training data required and to improve the accuracy of the results. Avenues of research in this direction are already appearing, as I now discuss.

**Unifying software engineering and DP.** We can use traditional software engineering techniques to evaluate and improve DP-generated code. DP systems already filter out incorrect programs by running test cases provided in the input and checking if the runtime exceeds some predefined limit.[8,18] Much more can be done such as performing code scans and pen-tests to find vulnerabilities, using bug-finding tools to detect errors, and profiling the synthesized code to find bottlenecks. This feedback can then be used to update the code to improve security and robustness. Not only will such capabilities benefit the specific synthesized programs, but it will also help fine-tune the underlying DP engine to improve code generation in the future.

**Integrating search-based program synthesis with DP.** An active area of research is *search-based program synthesis*[5] which, in its simplest form, uses brute force to search the space of

a Another issue to be dealt with is how DP will guard against malware found in code repositories. One can imagine nefarious actors purposely populating repositories with malicious code to be used for training models and thereby causing DP to produce insecure or even harmful code. There are multiple defenses against this, such as using DL to detect and remove malicious code from repositories, but this needs more exploration before programs produced by DP will be trusted.

b DP actually suffers from many of the same issues that plagued widespread adoption of MDD: careful specification of the problem is required, a rigorous process must be enforced (in the case of DP, continuous training of the model), and the difficulty of managing the lifecycle of the generated code.The latter spawned much work on how to maintain consistency between the code and the model. This has yet to be addressed by DP.

possible programs that satisfy specific syntactic and semantic constraints. While this is computationally expensive, searching an exponential space of potential solutions, techniques are being developed to make it more scalable.[3] Using DP for rapidly generating snippets of code and using search-based synthesis for integrating these snippets would extend the reach of DP code generation on the one hand and scale search-based program synthesis on the other. One recent work[15] pursuing this approach uses large pretrained language models (such as GPT-3 and Codex) to find small Python programs invoking Panda APIs that meet a textual description of the desired behavior, and then uses program analysis and synthesis, driven by input/output examples, to correct syntactic and semantic errors in the DP-generated program. Additional breakthroughs are needed in program synthesis to make these techniques viable for large real-world programs.

**Combining symbolic reasoning with DP.** A new area of AI research is Neuro-Symbolic AI,[21] which combines a rules-based approach with DL. One work that demonstrates its potential benefit combines DP with Inductive Logic Programming (ILP)[12] to extend ILP to deal with noisy data. More speculatively, one can use symbolic reasoning to complement DL and synthesize programs. For instance, say we have a Baysian-like network that can infer desired states in a house such as "if it is nighttime and it is not the bedroom then lights should be on when someone is in the room (95%)." When installing sensors in a smart home that can detect "people are in the room" and smart lights that can be turned on and off digitally, the system could reason about the desired state using the Baysian network and use DL to create a program to turn on the lights when someone enters the room and turn them off when he or she leaves.

For software engineers and AI researchers, many opportunities exist to leverage DL for synthesizing code and building more robust programs. It will require new paradigms that go beyond transformer-based DP systems. It will most likely become a collaborative effort, where the AI suggests, clarifies, and challenges the programmer. It will learn from multiple sources (such as Stack Overflow, code reviews, change management systems, agile standups, and even user manuals) and not just from code repos. Based on what we know today, programmers need not worry about their jobs. DP will not replace programming. Its aim should be to increase the productivity of software development and thereby make up for the significant shortage of programmers today.[22] C

**References**

1. AI Will Replace Coders By 2040, Warn Academics; https://bit.ly/3FyfQM9
2. Aiken, A. Programming Tomorrow's Machines," PLDI; https://bit.ly/3uXOJFi
3. Alur, R. et. al. Search-based program synthesis. *Commun. ACM 61*, 12 (Dec. 2018), 84–93.
4. Austin, J. et al. *Program Synthesis with Large Language Models*; https://bit.ly/3Fygc5r
5. Bodik, R. and Jobstmann, B. Algorithmic program synthesis: Introduction. *International Journal on Software Tools for Technology Transfer 15*, (2013), 397–411.
6. Boh, W.F. and Yellin, D.M. Using Enterprise Architecture Standards in Managing Information Technology. *Journal of Management Information Systems 23*, 3 (2006), 163–207.
7. Brooks, R. A human in the loop. *IEEE Spectrum* (Oct. 2021), 48–49.
8. Chen, M. et al. Evaluating Large Language Models Trained on Code. arXiv, July 2021; https://bit.ly/3j40AyR
9. Crofts, N. Whatever happened to model driven development? (Oct. 2020); https://bit.ly/3uYo2k4
10. Devlin, J. et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, In Proceedings of NAACL-HLT, Minneapolis, MN, 2019.
11. Dijkstra, E.W. On the foolishness of `natural language programming'; https://bit.ly/3V5ZP5J
12. Evans, R. and Grefenstette, E. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research 61*, 1 (2018), 1–64.
13. Founder Effect; https://bit.ly/2LxOwjK
14. GitHub Copilot; https://bit.ly/3YIdxPx
15. Jain, N. et al. Jigsaw: Large language models meet program synthesis. In *Proceedings of the 44th International Conference on Software Engineering*, (2022), 1219–1231.
16. Karpathy, A. Software 2.0. *Medium*, (Nov. 2017); https://bit.ly/3uYkdv6
17. Kastner, C. Machine learning in production/AI engineering; https://bit.ly/3FAv5US
18. Li, Y. et al. Competition-level code generation with AlphaCode. arXiv (2022); https://bit.ly/3hzrka4
19. Lu, S. et al. CodeXGLUE: A machine learning benchmark dataset for code understanding and generation. In *Proceedings of the NeurIPS Datasets and Benchmarks 2021*.
20. Ruchir, P. et. al. CodeNet: A large-scale AI for code dataset for learning a diversity of coding tasks. In *Proceedings of the NeurIPS Datasets and Benchmarks 2021*.
21. Susskind, Z. et al. Neuro-Symbolic AI: An emerging class of AI workloads and their characterization. arXiv (2021); https://bit.ly/3BHnRxk
22. The software developer shortage in the U.S. and the global tech talent shortage in 2022. DAXX (Jan. 2022); https://bit.ly/3uZ9rVu
23. Thompson, N.C. et al. Deep learning's diminishing returns. *IEEE Spectrum 55*, (Oct. 2021), 51–55.

**Daniel M. Yellin** (daniely@il.ibm.com) is Vice President, IBM Cloud, and IBM Distinguished Engineer at IBM in Armonk, NY, USA.

I would like to thank Michael Elhadad, Assaf Marron, Vugranam Sreedhar, Mark Wegman, and Natan Yellin for comments on an earlier version of this Viewpoint. I also would like to thank the referees and editors who made valuable suggestions improving the content and presentation of this Viewpoint.

DOI:10.1145/3571279

Juan E. Gilbert, Jeremy A. Magruder Waisome, and Simone Smarr

# Viewpoint
# An Analysis of Black Faculty in CS Research Departments

*Exploring Black faculty at computer science research departments where Ph.D. programs exist.*

Most ACM members reside outside the U.S., and while diversity issues vary around the world, we thought it would be enlightening to do a case study on one marginalized group in the U.S. in the hopes that the lessons learned could be helpful to other groups and in other regions. This case study is on the education origins of Black faculty members in computer science (CS) at U.S. universities. (Note: We use the term "Black" to represent "Black/African-American.")

Black tenure-track faculty members are severely underrepresented in computing research departments. According to the most recent Computing Research Association (CRA) Taulbee Survey, there are a mere 83 (1.7%) Black CS-tenured and tenure-track faculty members at U.S. Ph.D.-granting institutions includes 18 (0.9%), 25 (2.3%), and 40 (3.1%) Full, Associate, and Assistant Professors, respectively.[5] Using U.S. Department of Education data,[5] the breakdown by academic rank as compared to that for Black professors across all degree-granting institutions, including Historically Black Colleges and Universities (HBCUs) is shown in Table 1.

IMAGE BY MAX4E PHOTO

According to the most recent U.S. Census, 13.4% of the U.S. population is Black,[2] so all the numbers in Table 1 represent significant underrepresentation. Worse, it is a situation at the faculty level is unlikely to change soon because while 10.1% of bachelor's degrees are awarded to Blacks, the CRA Taulbee Survey reports a mere 203 (1.4%) of the students currently enrolled in CS Ph.D. programs are Black.[5]

In this Viewpoint, we look at the Black faculty currently in CS research departments (meaning those with a CS Ph.D. program), asking:

- Where are the Black faculty?
- Which research institutions have the most Black CS faculty?
- Which undergraduate alma maters supply most of the Black faculty? HBCUs or other institutions?
- Which graduate school alma maters supply most of the Black faculty?

**Table 1. Black professors across U.S. institutions and CRA institutions.**

| | Assistant | Associate | Full |
|---|---|---|---|
| CRA CS data for research institutions | 0.9% | 2.3% | 3.1% |
| DOE data across all institutions | 3.4% | 5.4% | 6.4% |

The CRA Taulbee Survey provides summative information about the demographics of the CRA member institutions. This information is very useful; however, it does not give specific details such as the number of Black faculty at each member institution, or the number of Black Ph.D. students at the institution level in CS. To illuminate these details, an analysis of Black faculty at CS research departments was conducted. CS research departments are defined as those that have a Ph.D. program.

Jeff Huang at Brown University used a class to do a crowdsourcing data experiment to collect information about CS faculty at 51 top research universities in the U.S.[3] This data shows that the top 11 producers of CS faculty in his study account for the majority (53%) of the entire CS faculty across those universities. Our research team did a similar data experiment, but this time the experiment was to look for Black faculty.

In December 2021, using the CRA Taulbee Survey's list of member institutions,[5] *U.S. News & World Report*'s graduate program rankings,[4] and the authors' knowledge of CS Ph.D. programs, 194 U.S. Ph.D. granting CS departments/colleges websites were searched with 152 (113 public, 39 private) from the 2020 CRA Taulbee Survey and the others from places the author identified, including HBCUs that have a CS Ph.D. program.

The search query was structured as follows: *institution name computer science faculty, for example, University of Florida computer science faculty*. During the search, the CS department or college website was browsed to identify the faculty listing. This exercise was done to visually identify Black faculty on the CS departments/colleges websites. If photos were not present, a search for the faculty members' personal websites and LinkedIn profile was conducted. During the search, the faculty member's current institution, their name, rank (Full, Associate, Assistant), gender (as identified by the author), undergraduate institution, and Ph.D.-granting institution were recorded. It should also be noted that some of the faculty identified did not have a CS Ph.D. degree. For example, there were faculty from Industrial Design and other areas. We did not make a distinction of the faculty members' Ph.D. discipline, only that they are currently tenure-track faculty in a CS research department or college.

**Black tenure-track faculty members are severely underrepresented in computing research departments.**

## Findings

Each of the major findings are described here.

**Black faculty.** 106 Black tenure-track faculty from 70 different institutions were found during the search. Therefore, 124 (63.9%) CS Ph.D.-granting departments/colleges do not have a single Black faculty member. The 2020 CRA Taulbee Survey reported 83 Black tenure-track faculty, but our search identified 85 at the Taulbee departments. The difference is likely related to timing. This search was done in December 2021, and the Taulbee data was collected earlier in 2020/2021. However, the difference of two faculty is minor and suggests this study's protocol was effective in identifying Black faculty.

Overall, this study found 32 (30%) Black female faculty and 74 (70%) male Black faculty, distributed among the academic ranks as shown in Table 2.

Howard University and the University of Florida (UF) had five Black tenure-track faculty, which was the highest number for any institution, see Table 3. Auburn University, DePaul University, Massachusetts Institute of Technology (MIT), Rochester Institute of Technology (RIT), University of California at Berkeley, and the University of the District of Columbia (UDC) all have three Black faculty in the computer science department or college. Some 16 other departments or colleges have two Black faculty. These 24 colleges and departments account for 60 (57%) of the Black CS faculty at research departments/colleges.

**The Producers: Graduate alma maters.** 22 (21%) of the Black faculty identified received a Ph.D. from one of the top 11 institutions from Huang's report. In his analysis, these institutions accounted for 53% of all the faculty,[3] however, they only account for 21% of the Black faculty at CS research departments/colleges.

The University of Florida (UF) has produced the most Black faculty at Ph.D.-granting CS departments/colleges; UF has produced seven Black faculty followed by Georgia Institute of Technology (Georgia Tech) and MIT, which have each produced six. Table 4 shows the number of faculty

**Table 2. Number of Black CS faculty by gender and rank.**

| | Assistant | Associate | Full |
|---|---|---|---|
| Female | 19 (17.9%) | 9 (8.5%) | 4 (3.8%) |
| Male | 24 (22.6%) | 24 (22.6%) | 26 (24.5%) |
| **Total** | **43** | **33** | **30** |

**Table 3. Number of Black faculty by institution with two or more faculty.**

| Institution | Number of Faculty |
|---|---|
| Howard U. | 5 |
| U. of Florida | 5 |
| Auburn U. | 3 |
| DePaul | 3 |
| MIT | 3 |
| RIT | 3 |
| U. of California at Berkeley | 3 |
| U. of the District of Columbia | 3 |
| Bowie State U. | 2 |
| Brown University | 2 |
| Carnegie Mellon U. | 2 |
| Cornell U. | 2 |
| Florida Institute of Technology | 2 |
| Georgia Tech | 2 |
| Indiana University | 2 |
| IUPUI | 2 |
| Northwestern | 2 |
| Pace | 2 |
| Tennessee State U. | 2 |
| UMBC | 2 |
| U. of Alabama | 2 |
| U. of California at San Diego | 2 |
| U. of Michigan | 2 |
| U. of South Florida | 2 |
| **Total:** | **60** |

**Table 4. Number of Black faculty by Ph.D.-granting institution with two or more graduates.**

| Ph.D.-Granting Institution | Number of Faculty |
|---|---|
| U. of Florida | 7 |
| Georgia Tech | 6 |
| MIT | 6 |
| NC State | 4 |
| Berkeley | 3 |
| Michigan | 3 |
| Clemson University | 2 |
| Columbia | 2 |
| Howard University | 2 |
| Princeton University | 2 |
| Stanford | 2 |
| UC San Diego | 2 |
| UW Madison | 2 |
| Vanderbilt University | 2 |
| Virginia Tech | 2 |
| Washington University | 2 |
| **Total:** | **49** |

**Table 5. Number of Black faculty by undergraduate-granting institution with two or more graduates.**

| Undergraduate Institution | Number of Faculty |
|---|---|
| MIT | 3 |
| Addis Ababa U. | 2 |
| Covenant U. Nigeria | 2 |
| Georgia Tech | 2 |
| Harvard University | 2 |
| Morehouse College | 2 |
| Northeastern U. | 2 |
| Stanford University | 2 |
| UMBC | 2 |
| U. of Alabama | 2 |
| U. of Lagos, Nigeria | 2 |
| U. of Washington | 2 |
| U. of West Indies | 2 |
| **Total:** | **27** |

by Ph.D.-granting institution for those that have at least two Black graduates that are now faculty. These 16 institutions account for 49 (46%) of the Black faculty at CS research departments/colleges. Howard University is the only HBCU to produce at least two Black CS faculty at research departments/colleges.

At the time of this study, there were 66 Association of American Universities (AAU) institutions.[1] The AAU members are America's leading research universities. As such, the members earn the majority of competitively awarded federal funding for research. 42 (63.6%) of the AAU Institutions have at least one Black faculty member in CS. 60 (56.6%) of the Black CS faculty received a Ph.D. from an AAU Institution.

**The Producers: Undergraduate alma maters.** MIT is the only institution that has produced three undergraduates that are now faculty at CS research colleges or departments, see Table 5. Twelve other universities account for two Black faculty at CS research colleges or departments. These 13 institutions account for 27 (25%) of the Black CS faculty at research departments or colleges. Morehouse College is the only HBCU to have produced two Black faculty at CS research colleges or departments. With respect to undergraduate institutions, there were three Black faculty members that did not list their undergraduate institution on their website or CV; therefore, the data for those three undergraduate institutions is not included.

## Conclusion

This research study explored Black faculty at CS research departments, meaning those that have a Ph.D. program. The analysis found 106 Black tenure-track faculty (43 Assistant, 33 Associates, and 30 Full Professors) from 70 different institutions. Huang's top 11 producers accounted for 53% of all the faculty,[3] however, they only account for 21% of the Black faculty at CS research departments/colleges. UF was the top producer of Black tenure-track faculty at research departments followed by Georgia Tech and MIT.

MIT leads the way for undergraduate degree recipients who went on to be faculty. Howard University is the only HBCU to produce at least two Black CS faculty at research departments/colleges. At the undergraduate level, Morehouse College is the only HBCU to have produced two Black faculty at CS research colleges or departments. Howard University and UF lead the way with five Black tenure-track faculty. This study points to leading institutions that were not in Huang's top 11 list. Therefore, this information is useful for those recruiting Black CS researchers and for those looking for a Ph.D. program

**MIT leads the way for undergraduate degree recipients who went on to be faculty.**

and/or Ph.D. advisor. Furthermore, this analysis will hopefully lead to CS departments/colleges becoming more transparent with their demographics data. This study was possible because of the uniqueness of the target population, meaning the ability to visually identify Black faculty.

**Study limitations.** This study relied on the visual appearance of faculty, data found in their online profiles and the authors' firsthand knowledge of Black/African-American identity to identify Black/African-American faculty. Furthermore, some identified faculty may not self-identify as Black/African-American. The gender was also identified by the author during the search from the visual appearance of the faculty member and their CV or LinkedIn data. Again, some faculty may identify differently than the author's classification in this study. These limitations were carefully considered during the search. The authors used these factors to count faculty within their best judgment. C


**References**

1. Association of American Universities, 2021; https://bit.ly/3FLElWf
2. Data.census.gov. Exploring Census Data, 2021; https://bit.ly/3FL0uE4
3. Huang, J. Analysis of Over 2,000 Computer Science Professors at Top Universities, 2021; https://bit.ly/3jleBsn
4. *U.S News and World Reports*. Best Computer Science Schools, 2022; https://bit.ly/3FRgXrH
5. Zweben, S. and Bizot, B. 2020 Taulbee Survey. *Computing Research News 33*, 5 (2021); https://bit.ly/3HQE6vK



**Juan E. Gilbert** (juan@ufl.edu) is the Banks Family Preeminence Endowed Professor and chair of the Computing and Information Science and Engineering Department at the University of Florida, Gainesville, FL, USA.

**Jeremy A. Magruder Waisome** (jam323@ufl.edu) is an assistant professor in the Department of Engineering Education at the University of Florida, Gainesville, FL, USA.

**Simone Smarr** (ssmarr@ufl.edu) is a Ph.D. student in the human-centered computing program at the University of Florida, Gainesville, FL, USA.

DOI:10.1145/3573127

Article development led by acmqueue
queue.acm.org

BY MATTHEW BUSH AND ATEFEH MASHATAN

# From Zero to 100

## Demystifying zero trust and its implications on enterprise people, process, and technology.

CHANGING NETWORK LANDSCAPES and rising security threats have imparted a sense of urgency for new approaches to security. Zero trust has been proposed as a solution to these problems, but some regard it as a marketing tool to sell existing best practices while others praise it as a new cybersecurity standard. This article discusses the history and development of zero trust and why the changing threat landscape has led to a new discourse in cybersecurity. Drivers, barriers, and business implications of zero trust provide a backdrop for a brief overview of key logical components of a zero trust architecture and implementation challenges.

In recent years, attackers have sown the seeds to feed a growing awareness of the flaws in common cybersecurity practices. Firewalls were applied to create a strong perimeter around enterprise networks; however, once inside the perimeter, an attacker can easily move through a company's intranet. With the increasing adoption of mobile and cloud technologies, a singular perimeter is becoming more difficult to enforce. With new attack vectors and technological changes, perimeter-based security models are moving toward obsolescence.

Despite the prevailing attitude that most threats are external, the source of attacks is approximately even between external attackers and insiders.[1] Insiders and supposedly trusted devices have become a serious cause for concern. Network access is being spread out among mobile devices, cloud users, employees working from home, and Internet of Things (IoT) devices. According to Cisco's 2018 Annual Cybersecurity Report, the scope of attacks is also increasing. Organizations reported a growing trend of security breaches that affected more than 50% of systems.[4] Larger breaches means larger sums of money are required to handle the aftermath. Security directly affects the bottom line, and high-level decision-makers need to take notice.

The approach wherein security teams have little input regarding high-level business decisions is outdated. The 2017 Equifax disaster, for example, forced the CEO, CIO, and CISO to resign. Capital One, a massive player in the U.S. financial industry, experienced a similar situation in 2019 with more than 100 million accounts compromised, followed by a huge class-action lawsuit.[5] The scale of cyberattacks, breaches, and privacy violations is getting larger and becoming increasingly visible to the public. Many organizations desperately need to improve their approach to cybersecurity, or this problem will only get worse.

Amid this changing cybersecurity landscape, zero trust has been lauded as a potential solution to many of these problems. With all the hype, though, many are unsure about what

IMAGE BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES


ZERO
TRUST
bersecur
irewalls

the term exactly means and how to implement it. Where did the concept of zero trust come from, and what does it mean for businesses today? Is zero trust a new standard or just cybersecurity done properly?

## De-Perimeterization and Zero Trust

The core ideas behind zero trust were first given major consideration in 2004. The Jericho Forum recognized that traditional security practices were quickly becoming inadequate because of increasing numbers of endpoints and device mobility requirements.[3] Firewalls, antivirus, and intrusion detection system (IDS) did not consider the threat from within the network. The initial term used was *de-perimeterization*, which focused on strengthening internal defenses and placing less emphasis on the external boundary. In 2007, the Jericho Forum published "commandments" that outlined principles and practices necessary for de-perimeterization.[10] The same year, DISA (Defense Information Systems Agency) and Department of Defense identified some key principles. The security strategy called "black core" was a security model that focused on moving away from perimeter security and focusing on individual transactions.[18] While the Jericho Forum and DISA laid a lot of the groundwork for de-perimeterization and shifting security focuses, they were not nearly as well-recognized as zero trust is today. It was not until John Kindervag from Forrester Research, Inc. introduced the concept of zero trust and zero trust architecture (ZTA) in 2010 that these ideas truly caught on.[11,12] The accompanying table illustrates a timeline of significant events in the history of zero trust conceptualization.

Zero trust is a broad concept that can apply to technologies, network architectures, and security policies. It has been described as a technology-agnostic mindset that puts security first. It requires that all actors within a network be treated as if they could pose a threat, because they really do. Enterprise resources should be protected individually and subjects accessing these resources should be constantly evaluated to ensure they are not a threat.

According to one of Forrester's seminal reports, three core concepts enable zero trust: All resources must be accessed securely; strict access control based on least privilege must be enforced; and all traffic must be inspected and logged.[12]

The problem with this conception of zero trust is that none of these ideas is particularly novel. In fact, they would strike most IT professionals as good security hygiene we should have been practicing all along. Many security professionals would agree that security should be a design goal of any enterprise system and that insider threats are just as dangerous as any others. This skepticism is warranted as many early explanations of zero trust fail to capture its unique value. A more recent definition from Jason Garbis and Jerry W. Chapman does a better job of explaining the core concepts enabling zero trust:

"A zero trust system is an integrated security platform that uses contextual information from identity, security and IT Infrastructure, and risk and analytics tools to inform and enable the dynamic enforcement of security policies uniformly across the enterprise. Zero trust shifts security from an ineffective perimeter-centric model to a resource and identity-centric model. As a result, organizations can continuously adapt access controls to a changing environment, obtaining improved security, reduced risk, simplified and resilient operations, and increased business agility."[9]

This definition draws attention to the difference between a zero trust *system* and the more abstract zero trust *mindset*. This mindset should not be confused with ZTA, which is any security architecture that enables zero trust. A zero trust mindset emphasizes that enterprise security should be extensible so any future additions support the goal of zero trust. Zero trust recognizes that no two security environments are the same. For example, data centers have unique security requirements when compared with cloud deployments or IoT networks. The National Institute of Standards and Technology (NIST) Special Publication 800-207 goes into further detail about potential implementations. It outlines multiple architecture models and technological implementations of the core logical components.[18]

In 2018, Forrester updated its original ideas for zero trust with the ZTX (zero trust eXtended) platform, which is intended to assist an organization's efforts in acquiring technology and implementing ideas for adding zero trust to legacy systems.[6] It is based on seven capabilities for mapping solutions to zero trust implementations: data; networks; people; workloads; devices; visibility and analytics; and automation and orchestration. This work ensures zero trust does not stagnate and that new ideas can be implemented within a zero trust environment.

**Significant events in the history of zero trust.**

| | | |
|---|---|---|
| 1 | 2004 | The Jericho Forum introduces the concept of "de-perimeterization" |
| 2 | 2007 | DISA and the Department of Defense identify "black core" security strategy<br>The Jericho Forum publishes its Commandments for de-perimeterization |
| 3 | 2010 | John Kindervag introduces zero trust |
| 4 | 2014 | Google publishes the first BeyondCorp article about its efforts to implement zero trust |
| 5 | 2018 | Forrester Releases the ZTX platform |
| 6 | 2020 | NIST publishes SP 800-207 providing guidance on zero trust |

## Why *Trust* Zero Trust?

The intersection of people, process, and technology (PPT) has long been a critical focal point for organizations evaluating their business practices. The PPT framework was introduced in the 1960s by Harold Leavitt in *Applied Organizational Change in Industry*.[14] The original model, which consisted of tasks, structure, people, and technology, combined tasks and structure into the process category, resulting in PPT. Bruce Schneier brought PPT to the forefront of the information security field in the late 1990s.[19] Even though it has been suggested that technology has become the most crucial aspect in today's threat responses, a strong relationship remains among all aspects of PPT. The organizational change that accompanies the transition to zero trust can be mapped onto this framework for a more structured approach to the topic.

Zero trust is a new mindset that

requires sweeping changes to be implemented effectively. There are obviously numerous drivers and barriers influencing an enterprise's choice to adopt zero trust, and so the question remains: Why *trust* zero trust?

**People.** The zero trust mindset brings several benefits that are motivating enterprises to consider adopting it. Zero trust requires that all employees take an active role in the security of an organization. With the increased prevalence of remote work, it is more important than ever that these workers be made aware of good security hygiene. An organization that commits to zero trust will likely try to increase organizational awareness of security and drive acceptance of zero trust.[17] This will have the knock-on benefit of forcing organizations to educate their employees about security. A more security-aware workforce will be a natural by-product of zero trust adoption.

From a cultural perspective, zero trust fosters interdepartmental cooperation and the adoption of IT. Since the network's systems will be geared toward secure networking, cooperation is critical to achieve a working zero trust security architecture. The successful integration of teams can improve network and security extensibility and prevent finger-pointing when problems arise.[5] Resources can be distributed equitably because these security and infrastructure teams will be cooperating and no longer competing to achieve different goals.

Not only might it improve the cooperation among existing teams, but zero trust also enables more flexible hiring. When it comes to hiring skilled workers, the ability of zero trust to create a superior remote work environment relieves security fears surrounding new hires. During a time where remote work is becoming increasingly prevalent, security and efficacy considerations may prevent an enterprise from hiring otherwise excellent employees.[20] Zero trust can help alleviate some of these fears and makes security onboarding much easier for remote work.

The final major people-centric driver of zero trust is management support. Senior decision-makers can be more easily convinced to take on zero trust initiatives because they can lead to long-term cost savings. Legacy networks often require a multitude of vendors, technologies, and solutions to ensure they remain secure. More time, money, and staffing are necessary to support these complex networks. Consolidating these controls into a more uniform and extensible zero trust solution means that vendor, management, and upkeep costs can be reduced more easily.[6] The increasing awareness that security is not just a cost sink, and that proper implementation can save time, money, and effort will make it much easier to garner support from upper management for zero trust projects. A successful zero trust endeavor will aid in digital transformation.[5]

**Process.** A unique opportunity to improve security posture has been presented to all enterprises affected by the pandemic. Zero trust drives enterprises not only to patch the holes created by transitioning existing processes, but also to improve the security of business processes. The reimagining of business processes will allow for better flexibility with remote work and a security-first mindset.

The open-ended nature of zero trust will enable greater extensibility to ensure organizations can freely adapt their security to the threat landscape. Extensibility by design is further aided by zero trust's emphasis on data collection and auditing. This allows enterprises to view information required to make informed decisions about secure process adjustments.[6] Collectively, this serves to reduce future costs and increase an organization's agility.

Security processes will likely be improved in the short term as well. Many zero trust deployments can take advantage of continuous authentication to reduce the friction that employees experience without compromising security. Google's BeyondCorp deployment effectively eliminated the need for a virtual private network (VPN) when remotely connecting to the network.[2] This saved both time and frustration related to configuring and connecting to a VPN. Research has found that 87% of security professionals who have adopted a zero trust approach have found that it improved productivity.[20]

Additionally, persistent data collection on the state and behavior of an entity allows for modeling normal entity behavior on the network. Continuous authentication allows security checks to identify unusual behavior and quickly respond.[7] This streamlines both everyday security processes and threat response processes.

One of the key tenets of zero trust is to protect data by collecting data. A working zero trust deployment requires an extensive audit of organizational resources. Knowing where different data is stored and what it is used for improves data organization and promotes informed security decision-making. Data can be identified and protected with varying degrees of security, depending on its sensitivity.[5] Better data management also aids privacy initiatives, such as ensuring General Data Protection Regulation (GDPR) compliance because the organization will know where its relevant data is stored and how to protect it.

**Technology.** Zero trust also has the potential to have a significant impact on the way enterprises use technology. Networks can be greatly simplified and split into modular segments, reducing maintenance and upgrade costs. Rather than continue managing the patchwork of devices and protocols across the whole network, individual segments can be implemented one at a time. This way the entire network need not be changed every time an upgrade is required.[11]

Segmentation also allows flexibility in different parts of the network. An IoT network may require a different security configuration than a network supporting cloud applications. Segmentation provides flexibility in implementing individual segments without compromising the overall needs of a zero trust network.[6] Industry regulations, such as Payment Card Industry Data Security Standard (PCI DSS) require only that the relevant segmented area meets all the specific regulations. A properly segmented network means that the focus on compliance can be limited to the segments that require it.[5] This again helps to reduce overall network complexity, saves money on compliance initiatives, and enables extensibility.

The technology that enables zero trust enables better security as well. NGFWs (next-generation firewalls) allow for inspection of application-layer traffic. Therefore, attack signatures that are otherwise invisible to traditional firewalls can be identified and quickly dealt with.[9] The damage from

attacks that do get through is limited because of network segmentation.[5] Overall, technology enables zero trust to enhance threat response and mitigate successful attacks.

Zero trust aims to combat threats by providing superior data protection. By protecting individual resources, granular data-protection rules can be implemented. As a result, large-scale data breaches will become less common because data access will be predicated on a risk-based approach. The location, value, sensitivity, and common usages of data will clearly define how it can be accessed. Network segmentation can further reduce the risk of individual breaches by keeping sensitive data in separate parts of the network.[7]

Trust through entity verification is not a new concept for cybersecurity professionals. Zero trust simply advances the idea by requiring that trust to be earned constantly rather than only once. BYOD (bring your own device) schemes bring inherently untrustworthy devices into the network. A universal set of trust requirements that apply to all devices makes managing security a much easier task.[7]

A zero trust mindset will have numerous benefits for any organization with significant digital assets. Properly implemented, ZTA will provide greater visibility into networks while simultaneously improving vulnerability management and breach detection.

Zero trust requires the inspection of all network traffic. NGFWs can inspect application-layer traffic, making it easier than ever before to get a clear picture of an organization's network. The added visibility helps to mitigate or even prevent data breaches. If unusual traffic can be inspected for signs of a breach, then it can be dealt with much faster. The FireEye M-Trends 2019 report states that the median time to discover a breach is 78 days.[8] Better network visibility means identifying vulnerabilities faster and allowing security professionals to react before anything has a chance to occur.[5]

## A Word of Caution

Given the number of factors driving adoption and the numerous benefits of zero trust, it would seem as if every organization should be rushing to implement a ZTA. Despite vendors complaining that zero trust is a game changer, it should not be implemented without caution. A variety of barriers and potential downsides means that zero trust adoption must be considered carefully.

**Organizational readiness.** Before an organization commits to zero trust, it must consider cultural compatibility. Zero trust security and network professionals are better-versed and more likely to see the value in the systems they already work with. If the teams responsible for implementation and operation are not convinced or run into too many difficulties during the transition, it will not be successful. Similarly, during implementation, user issues can diminish support for zero trust and lead to the perception that it decreases productivity and increases frustration.

It will be challenging for any organization to reap the benefits of zero trust if its users do not understand its value. For example, WestJet found it necessary to establish CoEs (centers of excellence) to promote learning and cultural acceptance of zero trust.[13] An enterprise that cannot provide similar resources may struggle to get networking and security professionals on board.

Since Kindervag's original work in 2010, a few organizations have begun implementing ZTA. A telling feature of this transition is that only recently have complete or near-complete zero trust initiatives been evaluated. Google's BeyondCorp is an example of a complete organizational shift that took multiple years from start to finish.

ZTA means a substantial pivot that requires forward-thinking, executive support, contingency plans, and commitment. Shifting to zero trust without a plan can result in half-complete measures and more complexity for security professionals to manage. Culture, technical bias, and emotional stake in current security practices can be a major barrier to success with zero trust. There may also be a misconception that zero trust is simply a marketing buzzword.[9] Therefore, organizations hoping to adopt a zero trust approach must be prepared for a long process with both political and technical issues that need to be resolved.

Zero trust is rarely being implemented by organizations that can start from scratch, but in its beta version of ZTA design principles, the U.K.'s National Cyber Security Centre (NCSC) acknowledges that even a greenfield environment requires patience and careful planning.[21] This means many organizations with established security practices and infrastructure may lack the agility and flexibility to simply start operating with zero trust. During the transition an enterprise may require changes to systems that cannot afford significant downtime, or a process component may not be readily replaceable with a zero trust counterpart.

Any organization that lacks flexibility to deal with such issues may find itself in a difficult position. The organizational inertia that current processes have and lack of one-to-one replacements for various process components can make transitioning to zero trust a difficult endeavor.

**Technological challenges.** Zero trust depends on excellent data management and information literacy. To segment a network and implement necessary controls properly, an organization must have excellent knowledge of its own data environment. Where does traffic need to go? Which resources are required for what roles? Where is high-priority data located? These are just some of the questions that need to be answered for a successful zero trust deployment.

NIST SP 800-207 suggests a complete audit of network components, data sources, actors, and enterprise assets such as managed devices.[18] An organization that lacks visibility into these areas is much more likely to implement an ineffective security system at great cost. At worst, it will reduce the effectiveness of workers. Setting up systems with the requisite visibility and analytical capability can require a large amount of money and effort, causing a significant barrier to adoption.[6]

IAM (identity and access management) is a necessity in any proper zero trust system. Poor management of user groups and disparate identity providers can lead to significant difficulties while implementing zero trust. Zero trust policies leverage identity attributes and user groups to grant access to resources safely and reliably. Zero trust could be the catalyst for improving IAM practices, but this still means that an enterprise needs to address these problems if it is to proceed with zero trust.[9]

Assets and technology used in ZTA

implementation may not be easy to migrate from or replace without excessive cost. Service providers could be highly specialized, meaning that if they experience issues, enterprise resources and business functions could be disrupted. Many core components of zero trust can be challenging to implement with current commercially available solutions.[21]

Artificial intelligence (AI) and software-based agents might be used for security purposes on an organization's networks. These agents, in turn, need to interact with various ZTA components. How these agents authenticate themselves is a largely unsolved problem for ZTAs. A non-person entity (NPE) may have a lower bar to entry, so attackers trying to gain access to a network may impersonate an NPE. In addition, the NPE may be an attack vector itself that could be manipulated into performing malicious actions.[3]

Data stored as a part of network monitoring could also become a target for attackers. This information would be useful for reconnaissance and planning future attacks. The management tools used to encode access policies are also potential points of attacks because they contain information about user and policy data.[18]

Policy engine (PE) and policy administrator (PA) are single points of failure for ZTA. Should the PE become compromised by a malicious or incompetent administrator, its rules could be changed to the point where they disrupt operations or create vulnerabilities in the system. A compromised PA could allow access to resources that would otherwise be off limits. To mitigate this threat, a policy could be housed in a secure cloud environment or exist in several locations. The blockchain may be a viable solution to managing PE policy.[18]

Much of the traffic on the network may be opaque to layer 3 network-analysis tools. Traffic may be coming from non-enterprise assets or simply be resistant to monitoring. As a result, different methods must be used on encrypted traffic, such as collecting metadata and machine-learning techniques.[18]

## Recommendations for a High-Risk, High-Reward Journey

Adopting a zero trust approach to cybersecurity is a high-risk, high-reward option for an enterprise. Correctly envisioned, zero trust offers a myriad of security improvements, an improved cultural mindset toward security, cost savings, and a highly extensible starting point for adding further enhancements. While the benefits are significant, it should be noted that zero trust needs to be an ongoing effort and transitioning to a zero trust approach can be a long and arduous process. Recommendations for an enterprise looking to adopt zero trust should follow the PPT template.

**Zero trust depends on excellent data management and information literacy. To segment a network and implement necessary controls properly, an organization must have excellent knowledge of its own data environment.**

**People.** Given that zero trust adoption is a long process, it is important that it has proper managerial and cultural support. A newer or more agile company may have an easier time aligning business objectives with the zero trust approach. An established enterprise, however, requires more effort to refocus managerial and cultural attitudes. Google ensured it had an effective technical support strategy to reduce user frustration during rollout. It also engaged with any impacted teams very early on in deployment.[21] WestJet created CoEs to enable its network and security teams to familiarize themselves with and learn the new technology.[13] These were also places where people could address concerns and suggest solutions during deployment.

Such strategies to gain the support of stakeholders are key for adopting any new technology or process. Lobbying important managerial positions by reinforcing the benefits of zero trust is one way to obtain the necessary support. Naturally, management will not be convinced unless there is a well-constructed plan.

Getting over political barriers can be eased by clearly enumerating the benefits of zero trust, finding an active executive champion, starting with minor projects that will show clear benefits, or actively seeking to work with networking or cybersecurity counterparts. Avoiding the misconception that zero trust will destroy or replace the infrastructure and architecture that others have worked hard to develop is a good way to ease interdepartmental and political tensions.[9]

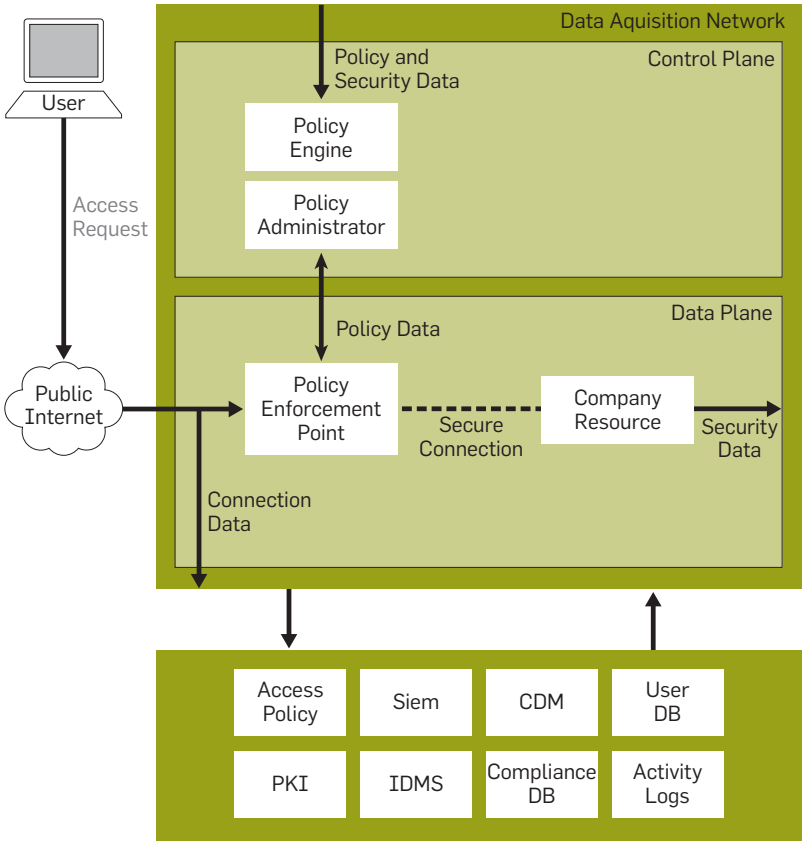
**Process.** The implementation process for zero trust is imperative for a successful transition. NIST SP 800-207 provides an effective way to transition from a perimeter-based network. The key steps are:

- Identifying actors who will use the system

# What Does Zero Trust Architecture Look Like?

**Although ZTA can be implemented in numerous ways, there are a few core logical components. The accompanying figure shows an example of a basic ZTA. A policy engine (PE) is located before the protected resources and makes the final decision regarding a subject's access to a given resource. A PE is paired with a policy administrator (PA), which is responsible for carrying out access decisions. It will signal to the policy enforcement point (PEP) that a session be created or destroyed.**

**Significant events in the history of zero trust.**




**The PEP acts as the gateway and manages the actual sessions between an entity and a resource. As these are logical components, the specific implementation details can vary, sometimes having a single device play multiple roles.[18] Many of these components also feed data into a data-acquisition network, which interacts with a variety of security policies, tools, and databases such as:**

- **Access policy**
- **SIEM (security information and event management)**
- **CDM (continuous diagnostics and mitigation) programs**
- **User databases**
- **PKI (public-key infrastructure)**
- **IDMS (integrated database management system)**
- **Compliance databases**
- **Activity logs**

**By interfacing with these and other data sources/sinks, the data-acquisition network allows for constant evaluation of the network's status. Apart from these core components, ZTAs can be implemented in various ways that have distinct advantages and disadvantages. For example, the UK's NCSC (National Cybersecurity Centre) suggests SSO (single sign-on) and token generation at the PEP, which is broadly applicable to many architecture models despite this not being a requirement for an effective ZTA.[15] Effective architecture is a moving target and ideas will change over time to support the core tenets of zero trust.**

- Identifying enterprise assets
- Identifying key processes and evaluating risks associated with executing process
- Formulating policies for the ZTA candidate
- Identifying candidate solutions
- Planning for initial deployment and monitoring
- Expanding ZTA

The requirement to perform extensive audits to gather information is a key takeaway. Successful zero trust enterprises need to know the impacted stakeholders, relevant assets, and the processes that must be reimagined during the transition. Otherwise, the organization is likely to have gaps in its implementation that can lead to even more significant security flaws in the future. ISACA has also identified visibility as a vital component of a successful ZTA.[17]

System analysis and auditing can be extremely difficult for complex systems. Zero trust deployments such as BeyondCorp and PagerDuty deployed zero trust broadly, spanning multiple business areas with fine-grained access control. They performed extensive network analysis to ensure these efforts would not interrupt productivity. This approach took much more time and effort than an incremental approach would. It can also be just as effective to onboard groups of users slowly and start with more coarse-grained access control. As the zero trust initiative progresses, it will become easier to tighten up these controls.[9]

Formulating policies, identifying potential solutions, and initial deployment should make use of all the collected information. Logical architecture and technological components should fit the enterprise's specific needs. The initial deployment and monitoring further collect information about how the implementation works and provide areas for improvement.

The final step is the most significant part of any implementation approach. Zero trust is meant to be extensible and provide the flexibility to adapt to changes in the security landscape. As ZTA is expanded, these steps should be repeated to maintain a continuously informed and improving security system. Given its novelty, zero trust networking will likely have evolving best practices, so a flexible approach is advised.[15]

A significant barrier to zero trust

initiatives is "analysis paralysis." Zero trust is a significant undertaking that contains many unknowns. An enterprise may also lack complete visibility into numerous internal factors. These problems balloon when approval from many stakeholders requires that a zero trust system immediately meet the same standards of existing enterprise systems. Consequently, zero trust initiatives can move extremely slowly and provide little to show for all of the effort. Effective milestones, close cooperation with stakeholders, and iterative deployment are some of the measures that can help reduce analysis paralysis.[9]

**Technology.** The final set of recommendations considers technology and implementation details. Zero trust is applicable in a variety of settings, including IoT, big data, and the cloud. The specific architecture should consider the environment for which it is being deployed. A company that is transitioning to zero trust may want to treat its IoT assets differently from internal applications. Network segmentation provides the flexibility to mix and match technologies and methods for different portions of the network. An enterprise should not attempt a one-size-fits-all architecture but instead, build out its zero trust network according to the requirements of different segments.

Many organizations are also subject to regulatory and compliance restraints. To avoid a situation where an enterprise zero trust system fails to meet compliance, it is essential to engage with external auditors and third-party compliance specialists early on.[9]

Additionally, ZTA should not be implemented with a singular focus on endpoints. An attacker able to subvert endpoints can move freely throughout a network again. Zero trust should be implemented in a layered approach that does not neglect network security policies. ISACA recommends implementing network security policies and adding endpoint capabilities to ensure the effects are complimentary.[14]

## Future Directions

As zero trust is deployed in increasingly complicated environments, security becomes more imperative. *Fog computing* occurs in a network where both users and devices are heterogeneous. There is an added caveat that server capacity in fog computing is far lower than in cloud computing, and as a result, zero trust would require a lighter-weight implementation.[22]

Next-generation network technologies will provide numerous benefits but will also cause more heterogeneity. Managing different types of access devices through the centralized architecture of zero trust poses a problem that is only beginning to be explored. Software-defined networking (SDN) is one solution that may fit well with zero trust networks.[22] Organizations such as Ericsson are aware that zero trust is technology-agnostic and have begun outlining how 5G networks impact ZTA.[16]

Zero trust requires that networks efficiently authorize traffic and data, but this becomes difficult as data volume and complexity increase. AI is one potential solution. By extracting features and classifying data, an efficient AI system could greatly increase the efficiency of a zero trust network.[22]

NIST is the only standards organization to provide a systematic exploration of zero trust. Literature about holistic schemes is rather thin outside of private organizations. As different architectures are developed for situations involving cloud networks, IoT, and big data, it will be necessary to classify general schemes that can achieve zero trust in different circumstances.

Zero trust brings few new security principles to bear, but more importantly provides an approach to get the most out of what cybersecurity professionals already consider good practice. Least privilege, strong authentication and access control, segmentation, defense in depth, and extensive logging and auditing are all existing practices that zero trust puts together with a cohesive goal in mind. The goal is security by design and a security-first mindset. Naturally, such a broad goal leaves much room for further research and development of new technologies that fit into ZTA. As new ways to enable zero trust emerge, security practices will only improve. As threats increase in number and severity, you should trust zero trust, because it includes all the things you should be doing anyway. C


**References**

1. Bendovschi, A. Cyber-Attacks—Trends, patterns and security countermeasures. *Procedia Economics and Finance 28* (2015), 24–31; https://doi.org/10.1016/S2212-5671(15)01077-1.
2. Beske, C. M., Peck, J., Saltonstall, M. Migrating to BeyondCorp: Maintaining productivity while improving security. *Login 42*, 2 (2017). Google Research; https://research.google/pubs/pub46134/.
3. Bleech, N. What is Jericho Forum? Visioning White Paper, Jericho Forum, 2005; https://collaboration.opengroup.org/jericho/vision_wp.pdf.
4. Cisco. *Annual Cybersecurity Report 8*, (2018), 19; https://www.cisco.com/c/dam/m/hu_hu/campaigns/security-hub/pdf/acr-2018.pdf.
5. Cunningham, C., Pollard, J., Holmes, D. The eight business and security benefits of zero trust. Forrester, 2019; https://bit.ly/3M9v2Cq.
6. Cunningham, C. *The Zero Trust eXtended (ZTX) ecosystem*. Forrester, 2019.
7. Embrey, B. The top three factors driving zero trust adoption. *Computer Fraud & Security 2020*, 9, 13–15; https://doi.org/10.1016/S1361-3723(20)30097-X.
8. FireEye. M-Trends 2019; https://content.fireeye.com/m-trends/rpt-m-trends-2019.
9. Garbis, J., Chapman, J.W. *Zero Trust Security*, 1st ed. Apress, 2021
10. Jericho Forum. *Jericho Forum Commandments*, 2007; https://collaboration.opengroup.org/jericho/commandments_v1.2.pdf.
11. Kindervag, J., Balaouras, S., Coit, L. *Build security into your network's DNA: the zero trust network architecture*, 2010, 1–26. Forrester; https://www.virtualstarmedia.com/downloads/Forrester_zero_trust_DNA.pdf.
12. Kindervag, J., Balaouras, S., Coit, L. No more chewy centers: introducing the zero trust model of information security. Forrester, 2010; https://media.paloaltonetworks.com/documents/Forrester-No-More-Chewy-Centers.pdf.
13. Kindervag, J., Mak, K. Case study: WestJet redefines its security with Forrester's zero trust model. Forrester, 2015; https://bit.ly/3fJgs8w.
14. Leavitt, H.J., March, J.G. *Applied Organizational Change in Industry: Structural, Technological and Humanistic Approaches*. Carnegie Institute of Technology, Graduate School of Industrial Administration, 1962; https://books.google.ca/books?id=P_KZNQAACAAJ.
15. National Cybersecurity Centre. *Mobile device guidance: network architectures* (2020); https://bit.ly/3rs6vin
16. Olsson, J., Shorov, A., Abdelrazek, L., Whitefield, J. Zero trust and 5G—realizing zero trust in networks. *Ericsson Technology Review* #05 (2021); https://bit.ly/3rrzFhA.
17. Pironti, J.P. Five key considerations when adopting a zero trust security architecture. @ISACA 7, 2020; https://bit.ly/3ygzAB8.
18. Rose, S., Borchert, O., Mitchell, S., Connelly, S. Zero trust architecture. NIST SP 800-207, 2020; https://csrc.nist.gov/publications/detail/sp/800-207/final.
19. Schneier, B. People, process, and technology. Schneier on Security, 2013; https://www.schneier.com/blog/archives/2013/01/people_process.html.
20. Sheridan, O. The state of zero trust in the age of fluid working. *Network Security 2021 2*, 15–17; https://doi.org/10.1016/S1353-4858(21)00019-2.
21. U.K. National Cyber Security Centre. Zero-trust-architecture. Github; https://github.com/ukncsc/zero-trust-architecture.
22. Yan, X., Wang, H. Survey on zero trust network security. *Artificial Intelligence and Security. Communications in Computer and Information Science* 1252 (2020). X. Sun, J. Wang, E. Bertino, Eds. Springer Singapore; https://doi.org/10.1007/978-981-15-8083-3_5.



**Matthew Bush** is a research assistant at Toronto Metropolitan University's Cybersecurity Research Lab. His research interests include managing privacy, security, and ethical concerns stemming from Internet of Things, big data, and machine learning.

**Atefeh Mashatan** is a Canada Research chair and an associate professor at the Ted Rogers School of Information Technology Management and the founder and director of the Cybersecurity Research Lab at Toronto Metropolitan University (formerly Ryerson University). She investigates challenges and opportunities brought forward by these new technologies and how they change the threat landscape of cybersecurity. In 2019, Mashatan was recognized by *SC Magazine* as one of the top five Women of Influence in Security globally.

DOI:10.1145/3573129

Article development led by acmqueue
queue.acm.org

A discussion with Michael Loftus, Andrew Vezina, Rick Doten, and Atefeh Mashatan.

# The Arrival of Zero Trust: What Does It Mean?

IT USED TO be that enterprise cybersecurity was all castle and moat. First, secure the perimeter and then, in terms of what went on inside that, "Trust, but verify."

The perimeter, of course, was the corporate network. But what does that even mean at this point? With most employees now working from home at least some of the time—often on their own smartphones or laptops—and organizations relying increasingly on cloud computing, there's no such thing as a single, enterprise-wide perimeter anymore.

And, with corporate security breaches having become a regular news item over the past two decades, trust has essentially evaporated as well.

John Kindervag, who articulated the zero-trust enterprise defense strategy a little over a decade ago, explained: "The concept is framed around the principle that no network, user, packet, interface, or device—whether internal or external to the (corporate) network—should be trusted. Some people think zero trust is about making a system trusted, but it really involves eliminating the concept of trust from cybersecurity strategy."

Easier said than done, naturally. Still, how well is that effort going? We asked Atefeh Mashatan, the founder and director of the Cybersecurity Research Lab at Toronto Metropolitan University (TMU), to explore that with three enterprise security professionals of long standing: Andrew Vezina, the CISO at Equitable (EQ) Bank; Rick Doten, the VP of information security at Centene Corporation, as well as CISO of Carolina Complete Health; and Michael Loftus, an IT consultant with considerable corporate security experience.

**ATEFEH MASHATAN:** There has been considerable buzz about the zero trust security model. Some call it the "gold standard" of cybersecurity, while others argue it's nothing more than marketing jargon for networking practices that organizations should have adopted some time ago. What's the truth of the matter?

**MICHAEL LOFTUS:** It's somewhere along that continuum since zero trust is a conceptual framework that challenges the long-held assumption that, if you're physically within a firewall perimeter and are part of some specific enterprise domain, implicit trust will be granted to both you and your device. Indeed, zero trust challenges the very idea that any sort of implicit trust can be considered safe any longer.

But if you're asking, "Don't these supposedly new zero trust measures often include techniques that have been deployed previously?", the answer is, "Absolutely!" This is particularly true when it comes to MFA (multifactor authentication) and advanced analytics. And yet, do I still think zero trust qualifies as something new and buzzy? Absolutely!

Ultimately, the truth is that zero trust is an approach that involves doing things differently with respect to how users and devices are authenticated,


IMAGE BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES

as well as how risk is managed. That's all because we will no longer assume that any part of the environment can be granted implicit trust.

**MASHATAN:** But shouldn't we have been doing this all along?

**LOFTUS:** I wouldn't say that, since I think zero trust came up in response to the many different lessons we have absorbed in moving to more distributed environments where applications, data, and identity mechanisms now reside in the cloud. With that shift, security mechanisms—once good enough to protect people on the enterprise premises who were linked to the enterprise network—became inadequate.

**MASHATAN:** Which techniques do you consider to be absolute minimum requirements for a zero trust environment? What other capabilities would you regard as highly desirable?

**ANDREW VEZINA:** Whenever I picture a zero trust architecture, I start in the middle and work my way out. The central element that everything seems to come back to is the trust engine responsible for making decisions about whether some particular user, coming from some particular device, ought to be allowed access to some particular resource or application. As Mike indicated, this used to be a much simpler call based on whether the user could supply the correct Active Directory password. But now, under zero trust, that's changing in two major respects.

First, we're going to be working with a much richer set of information when

MICHAEL LOFTUS

**The truth is that zero trust is an approach that involves doing things differently with respect to how users and devices are authenticated, as well as how risk is managed. That's all because we will no longer assume that any part of the environment can be granted implicit trust.**

it comes to users and their authentications. The biggest change here is the shift from a simple password to multifactor authentication. We're also starting to see many new capabilities and tools that will be able to take advantage of all manner of telemetry related to a user's login characteristics.

The second major improvement worth noting is the ability to do more in terms of authenticating devices simply because we're able to obtain more information about the devices people typically use whenever they attempt to access an environment. The move toward implementing zero trust starts with pushing all the authentication decisions out toward the apps such that this greater context might be incorporated in deciding whether access ought to be granted to whatever resource has been requested.

From there, you can start thinking about how people are going to access the applications and what this suggests your network ought to look like—bearing in mind you will no longer need a local area network or a corporate network to serve as a method of trust, which is to say we are moving to a security model where it basically doesn't matter where the network user is coming from since you will be relying on user and device telemetry for authentication instead of what the network tells you.

There's also an opportunity to simplify and improve the user experience through SASE (secure access service edge) and similar technologies. Fundamentally, this just gives users a cleaner, easier route to get from wherever they happen to be to whatever applications they're looking to use. That, in turn, leads to an architecture where you rely on SASE technology that's typically cloud-based to provide access to the Internet in a safe manner even as you also rely on software-defined perimeter technology to allow connectivity from the Internet to whatever application or resource a user may have requested.

**MASHATAN:** Now that you've given us a sense of how SASE can be combined with zero trust, can you compare the two to point out their similarities and differences?

**VEZINA:** First, I'd describe zero trust as a strategy, whereas SASE is a technology that, over the past few years, has largely displaced a whole class of on-prem tools many organizations once used to evaluate user traffic going out to the Internet—their egress traffic, in other words. Typically, that included Web content filtering, malware analysis, and perhaps even data-loss prevention analysis, along with a couple of other capabilities.

Prior to SASE, everything was built on the corporate network. So, if users were working at home and wanted to access the Internet for any reason, they'd first have to log onto the corporate network, do a VPN (virtual private network) connection to enter the network, and then execute a hairpin turn to go back out to the Internet—passing through the corporate proxy and malware tools, as well as probably three, four, or five different security tools along the way. With SASE, instead of requiring users to jump onto the corporate network first, they're now routed through a SASE provider where all the necessary security functions are provided by way of a cloud-based model that is complementary to the zero trust strategy in the sense that it doesn't require a corporate network.

Because these two concepts are complementary, it seems most of the organizations that are implementing zero trust architectures now are moving toward SASE as well. With that said, the two are not necessarily interdependent. That is, you can implement SASE and yet still employ a legacy model for security. And it's at least theoretically possible to implement zero trust without SASE, but that might prove to be difficult at this point, given current technology.

**RICK DOTEN:** I tend to look at SASE as infrastructure, which, as Andrew indicated, replaces a variety of tools previously operated on premises. And yes, putting all that in the cloud certainly simplifies things for users. That infrastructure also still provides a central point for implementing all policy decisions. That's all great, particularly for small and mid-sized organizations that don't have a lot of infrastructure and may already be relying largely on cloud-based applications and services anyway. For them, SASE is ideal.

But I represent a very large company where this would be regarded as sacrificing visibility and control since it doesn't allow for directly managing and configuring all the different fea-

ture sets and monitoring options, even if we are allowed to set the rules and control things at a high level.

Bottom line, SASE is great for small to medium-sized organizations that would love to rely upon a specialized external infrastructure to handle all their security monitoring, response, and controls. That certainly promises to reduce their own management burdens and basically gives their users a clean pipe to all their applications.

And yet, this isn't necessarily going to work well for all organizations. Besides the matter of scale, there also are certain regulated industries (particularly in the U.S.) where companies may not feel comfortable outsourcing protected data—such as healthcare information and client financial information—to a SASE provider. In this respect, it's important to remember there are certain things the SASE provider will want to look at in de-encrypted form, particularly when it comes to behavior monitoring.

It seems zero trust might be best described as a strategy or approach, which is to say it's somewhat nebulous and hard to pin down. That, of course, makes it an absolute gift to those who market cybersecurity products and services.

Indeed, zero trust has been promoted with real gusto. The trade press commonly uses the word "hype" in reporting about zero trust and the efforts made to market it. Yet there are aspects of the approach that even critics readily agree are entirely sensible.

One thing seems certain: The jury is still out as to whether zero trust will ever manage to live up to all the marketing froth.

**MASHATAN:** To date, it's fair to say the zero trust strategy has taken hold chiefly in North America. You don't see a lot of zero trust initiatives being launched in Europe, for example. Also, it's noteworthy that NIST (National Institute of Standards and Technology) is the only standards body so far to have released a guide for zero trust architecture.

Why is it that all this energy seems to have been concentrated in North America?

**DOTEN:** That's an insightful question since it points out that zero trust is being aggressively marketed to companies in the U.S. and not so much to everyone else. That does make you wonder if something might be going on here. What I think is at the root of this is a history in the U.S. of selling things through the subtle manipulation of fear, uncertainty, and doubt—the so-called "FUD factor." In this case, the pitch runs something like: "Here's this incredibly complex problem that's too big for you to handle on your own. But, for a price, we can help." As North Americans, we've been conditioned to respond to that sort of marketing, but it's not necessarily how the rest of the world has been conditioned. In fact, in many European countries, this sort of approach is illegal.

I do think zero trust is being hyped in the U.S., and, frankly, I'm embarrassed for our industry, as well as for NIST. In fact, if you look closely at the text in the NIST guide, you will find arguments that closely echo narratives already put forth by each of the big vendors—sometimes using the very same words.

I've been thinking quite a bit about this recently because, on the current roundtable circuit, there's a lot of talk about digital transformation and the push to move everything into the cloud. To be honest, the whole zero trust pitch just dovetails a little too neatly with that. In fact, I think zero trust is little more than the security field's tagalong concept to digital transformation.

The funny thing is, this stuff isn't even particularly new. I mean, when you think about it, we've had network access control for quite some time. As much as 20 years ago, you could connect to a VPN that would scrutinize your environment to make sure your operating system was up to date, your antivirus protection was on and attached, and you had the right versions of software loaded, and so on. All these things involved remote access. We also had behavior monitoring and risk-based monitoring.

These concepts are not new. The only difference is that, whereas we used to access all these applications through a portal in the corporate infrastructure, we now can get to them in the cloud without going through the corporate infrastructure—which is great but not exactly the huge leap forward that the zero trust marketing might lead you to believe.

**MASHATAN:** Do the rest of you agree the zero trust strategy is being oversold?

**VEZINA:** I share much of Rick's skepticism. Over the years, we have certainly seen plenty of new approaches to security, whether in the form of technologies or strategies. The introduction of malware sandboxes about 15 years ago as a new way to identify malware is one that comes to mind. Until then, we all used signature-based antivirus detection, which wasn't working so well. Then this new technology came out that proved to be a clear improvement. But it also was largely oversold as the solution to breaches and, accordingly, made billions of dollars for a few vendors. Now, it's nothing more than table stakes for most organizations, and it never actually managed to solve the problem. It helped in some cases, but then the attackers evolved.

It could be argued that zero trust is a bit different in that it's a strategy rather than a tactical countermeasure. But we don't know yet whether it's going to live up to the hype, and red team exercises will likely be one of the best ways to evaluate its effectiveness.

In fact, in recent years, I've been involved with several different red team exercises run against conventional defenses. Each of those attacks had different objectives and used different approaches. But the common theme was that, once the attackers established a foothold in the environment, there were many different avenues they could take to move laterally to get at whatever important asset they were looking to compromise or use while executing some sort of objective—like data theft or ransomware.

In the most recent red team exercise at our firm, we provided the attackers with two initial footholds in the environment. Machine #1 was a traditional workstation connected to the corporate network that was accorded the trust that commonly comes along with this. Machine #2 was more aligned to the requirements of zero trust in that it had no ability to connect to the corporate network or to any other machines on the network. Instead, it could be used to connect with and make use of only a small number of applications.

When initiating their attack from this machine, the attackers found they

were trapped in that user's workspace with no way to travel laterally. In this case at least, the decision to step away from trusting the corporate network limited the potential to move laterally, which kept these attackers from achieving their objectives.

Conversely, with machine #1 and its legacy security architecture, the attackers were easily able to move laterally to other machines and user accounts on the corporate network, and this ultimately allowed them to achieve their objectives while the SOC (security operations center) scrambled to respond to the attack. Given this result, I think an architecture designed to curb lateral movement is one that holds considerable promise.

**DOTEN:** I agree wholeheartedly. We used to joke that, while the corporate network might be hard and crunchy on the outside, it usually proved to be soft and chewy on the inside. There's a good argument to be made for any sort of defense that promises to reduce an attacker's ability to move laterally.

I also applaud your malware sandboxing example. The big lesson there is that people trusted that claim implicitly. I had a malware sandbox eight years ago and it worked 60% of the time exactly as advertised. The other 40%, it didn't. I mean, it didn't open the malware, it didn't block it … it just dropped it into the mailbox, which just goes to illustrate the hazards of returning to a vendor-driven environment where you buy something, install it, and then just simply expect it to work. The real danger is the false sense of security that you don't really need to do anything more about the situation since you're already protected. Guess again!

**MASHATAN:** People must find all the hype in this space disturbing. What telltale signs should we be looking for?

**DOTEN:** We are certainly not talking about anything quite like blatant lies and distortions. I think it's more a matter of vendors not giving customers all the help they need to filter through the glittering generalities in the marketing pitch. Mind you, zero trust as a concept is a perfectly fine approach. But if it doesn't happen to be something that matches up well with your organization's risks and processes, then it's not for you.

**VEZINA:** I think the best way to speak to this is by way of example. Some vendors push identity governance as the key to implementing a zero trust strategy.

One of the more prominent controls in identity governance is the practice of reviewing or recertifying user access on a certain frequency or based on certain events. It's every auditor's favorite security control. But I find it difficult to connect this with the core tenets of zero trust, whereas other identity controls such as MFA and the central trust engine are clearly aligned with it.

Whatever else might be said about zero trust, it isn't magic and won't be achieved with just a few simple steps. As a sea change in the overall approach to enterprise security, it's bound to entail considerable effort, substantial investments, and plenty of time—regardless of what the marketing might suggest.

Yet, for all that, maintaining a more traditional approach isn't a viable option since corporate attack surfaces now have grown much larger, the stakes are higher, and the threat models have evolved in a multitude of ways from those encountered just a few years ago.

So … what to do?

**MASHATAN:** Let's dive a bit deeper into the matter of zero trust implementation. Just how complicated is that?

**LOFTUS:** How complicated? That's probably in the eye of the beholder. The reality is that this represents a significant technology shift, which never is going to be trivial. Whether you're focused on authentication or looking to make changes to your network and move to a SASE provider, you've got some hard work ahead, and that's going to take some time.

Much of what is happening in the marketplace now is focused on the authentication side of things—MFA, cloud-based IdPs (identity providers), and application authentication protocols. That's just because all the things we've done over the years with active directory, Kerberos, and Windows NTLM (New Technology LAN Manager) just aren't going to cut it in a zero trust world. And then, of course, SASE is also exploding.

**MASHATAN:** To what degree do legacy systems and processes complicate matters?

**LOFTUS:** It's sure going to be difficult to get very far down this path if you're relying on old-school VPNs and on-premises directories and storage. To my way of thinking, you really need to have a cloud-enabled IdP, along with the ability to egress your traffic just as close to the end user as possible.

The real problem with all that legacy, however, is what it does to hamstring your ability to take advantage of the whole zero trust approach. That is, if you have a lot of legacy applications, you might be able to adopt zero trust to better control end-user access, but there also are many zero trust practices you won't be able to employ simply because you won't be able to intercept most legacy protocols.

**MASHATAN:** Once an organization has made the decision to adopt zero trust, what are its biggest challenges?

**VEZINA:** If that organization happens to be running legacy applications, as Mike just suggested, it's likely to face some significant difficulties, from both a zero trust and a cloud migration perspective.

On the user side, there's plenty more heavy lifting to be done. For one thing, you will need to make sure you've got the right technology in place to handle user authentication. And you are probably going to want to provide for that in a centralized manner that handles most, if not all, your applications. This means integrating all those legacy applications to the centralized identity provider.

So, there will be a fair amount of work to do, but—apart from dealing with legacy technology—I don't think of it as exceptionally challenging work. Still, there's no question but that it represents a lot of heavy lifting, and you'vee got to identify the right spots in your transition roadmap to make those investments.

**MASHATAN:** What might some of the early steps in that roadmap look like?

**DOTEN:** It goes back to the fundamentals of automating any business process—understanding what's there already, mapping out where you need to go, and then determining where your security gaps are likely to surface. In that respect, the challenges are already familiar. But then add to that the issues associated with legacy versus modern,

which we've already touched on. Then there's also the matter of scaling, which isn't limited just to the number of units since you will also need to account for all the new applications, all the things that come about owing to merger-and-acquisition activity, all the new users who come in—while noting that this all is inherently dynamic.

**MASHATAN:** Let's say the organization has made good purchase decisions and has a good zero trust roadmap. What are some of the implementation challenges it is still likely to face?

**VEZINA:** Most of the organizations that falter will do so because of incomplete implementations. An organization could easily select the wrong identity provider that doesn't necessarily offer a full range of capabilities—or, more likely, falls short by integrating only 20% of the company's applications, thus leaving all the others stuck in the old model. If that proves to be the case, it's going to be necessary to keep the old corporate LAN around, meaning the company is going to end up with a half-and-half architecture—which can prove really challenging.

The second area where I think difficulties will arise is on the endpoint side. That is, it's going to be necessary to have the right agent or two—or maybe even three—to provide telemetry from the endpoint up to the identity source. An investment in EDR (endpoint detection and response) capabilities will probably be required for many of the detection functions to relay some sort of health telemetry to the centralized decision-making engine. All of this is difficult.

There's also a good chance most organizations don't currently have agents on their endpoints capable of handling this, which means they're going to need to switch out their core endpoint technology for more modern capabilities, and that in turn means all their workstations will need to be updated to the right operating-system level.

So, right there, you have two significant challenges. I would describe both as "ground-floor issues" that need to be addressed just to make decent progress even possible.

That still leaves what I consider to be unsolved problems. One of those is: How is this architecture supposed to work for privileged users? How do they fit into this picture? Typically, they do a lot of their work on the corporate network. In fact, there are some proven session-management capabilities that were implemented in the past with this in mind. How are those capabilities going to fit into this new architecture? Somebody might already be working on this problem, but I don't have a clear sense of what that solution is going to look like.

Similarly, I don't know what sorts of provisions are going to be made for developers—who have special requirements of their own. That is, they too often work on the corporate network and tend to have unusually high performance needs. There definitely are some problems that remain to be solved.

**DOTEN:** My concern, speaking as someone who has been quite focused on application cloud security for a long time, is that zero trust so far has been approached almost exclusively as a technical problem. The reality is that it's just as important to understand the business requirements. IT security isn't simply about protecting IT; it's about protecting the business. What I think we're lacking at this point is a governance process that links the people who run the business with those who are responsible for the technical operations such that they can work together to ensure that what comes out of these zero trust efforts actually ends up addressing the resiliency, security, and privacy requirements of the applications and data on the business side of the house.

It's also true that engineers don't always think about the user experience or fully appreciate the limits on customer tolerance for lag time or the number of steps in some given process. This, too, is where input from colleagues with more customer contact can prove useful. The same might be said for reducing risk since requirements checklists are no substitute for what can be learned from road-tested experience.

**MASHATAN:** Can you draw on your own experience to point out some of the specific implementation challenges people might encounter?

**VEZINA:** The first thing that comes to mind—at least for larger organizations, given their scale and the amount of data involved—is that it can be hard to obtain the investments required even to start moving in a zero trust direction. Generally, though, there already

RICK DOTEN

**I do think zero trust is being hyped in the U.S., and, frankly, I'm embarrassed for our industry, as well as for NIST. In fact, if you look closely at the text in the NIST guide, you will find arguments that closely echo narratives already put forth by each of the big vendors—sometimes using the very same words.**

ANDREW VEZINA:

**One of the more prominent controls in identity governance is the practice of reviewing or recertifying user access on a certain frequency or based on certain events. But I find it difficult to connect this with the core tenets of zero trust, whereas other identity controls are clearly aligned with it.**

are plans in motion to make certain improvements aimed at keeping the infrastructure current. Often, at least some of that will align reasonably well with what also needs to be done to better secure the environment.

For example, maybe your network egress solutions are coming up for renewal, meaning you might also look to move all that to a SASE model. Or maybe there are certain applications that are pushing the organization to move to the cloud. As part of that, you could also look at re-architecting the authentication process so it fits better with the zero trust strategy.

This is just to say that it's going to take some strong leadership to bring the zero trust strategy to the forefront and start lining up all the necessary investments. Frankly, I think it's a given that some organizations will fail at this, which can happen even if all the concepts are understood and the company is motivated to move forward. None of that will matter if things can't be implemented in a timely manner.

**DOTEN:** One of the biggest challenges here is also one of the oldest challenges: intra-organization politics. When it comes to implementing something as all-encompassing as a zero trust initiative for a large company, you're talking about coordinating the efforts of a dozen teams or more. I mean, there's the identity team, the endpoint team, the application team, the VPN team, the cloud team, and on and on.

So, collaborating, integrating, and orchestrating becomes the name of the game. Getting these different groups to cooperate and move in the same direction can be enormously challenging just because you've got all these different fiefdoms to account for and you always discover some number of people who simply aren't ready to embrace change. Dealing with these sorts of issues generally proves to be a lot harder than handling the technical problems.

**VEZINA:** But now, if we shift to look at the other end of the spectrum, where you find all the smaller—possibly younger—firms that are less burdened with technical debt, we find a very different set of problems. Many of the issues we just discussed will not be concerns there. Instead, there might be problems executing details of the architecture. For example, the people whose job it is to assign access might simply be granting people full administrative access to the whole tenant space at a cloud provider. Or maybe they're adding tools to their stacks that are going to start looking for certain capabilities immediately—but, while those things happen to be there, nobody ever enabled them. Or maybe they're expecting their SASE provider to perform certain functions but neglected to turn on the necessary inspection capabilities.

This is only to say a lot of these execution failures plague many smaller organizations since they tend to have fewer specialists or maybe only a small set of generalists on staff. They're just going to be more prone to missing some of these details.

**MASHATAN:** Any advice on what not to do?

**DOTEN:** Not to suggest that you shouldn't listen to your vendors, but don't expect complete candor. This speaks to the point Andrew just made about smaller, less mature organizations: Since they lack relevant experience, they're going to end up relying more on their vendors. That can be a problem. The vendors are going to try to convince you that if you buy their particular platform, you'll achieve zero trust. But it doesn't work that way since this is more of a journey. There are maps to help you steer in the right direction, but there aren't any measurables to let you know whether you've arrived or not. That's just something you'll need to figure out by living with your system.

You need a plan and some clear guidelines. Don't set off on this journey without those. Still, it's fine if the plan is a simple one. In fact, we like to say the best way to eat an elephant is one bite at a time. So, just start off with one thing. Particularly if you're a small organization, keep your focus on that alone. Just make sure you've got everyone on the system identified and have MFA on everything. Then you can expand from there. But do it in steps; don't try to do it all at once.

One more thing: Don't set off on this journey until you have a clear set of business requirements in mind. Otherwise, the vendors will try to shape that agenda for you—and that probably won't lead to a happy outcome. C

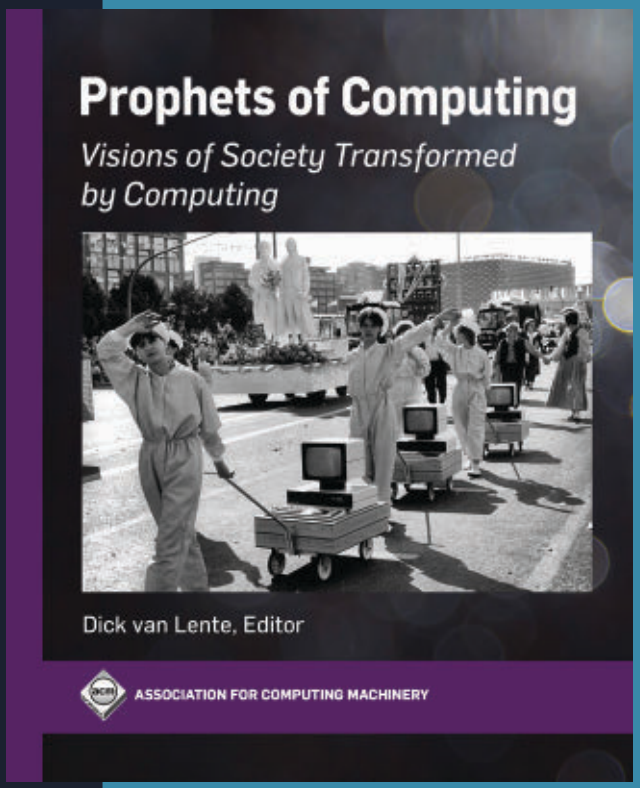
# ACM BOOKS

**Collection II**

When electronic digital computers first appeared after World War II, they appeared as a revolutionary force. Business management, the world of work, administrative life, the nation state, and soon enough everyday life were expected to change dramatically with these machines' use. Ever since, diverse prophecies of computing have continually emerged, through to the present day.

As computing spread beyond the US and UK, such prophecies emerged from strikingly different economic, political, and cultural conditions. This volume explores how these expectations differed, assesses unexpected commonalities, and suggests ways to understand the divergences and convergences.

This book examines thirteen countries, based on source material in ten different languages—the effort of an international team of scholars. In addition to analyses of debates, political changes, and popular speculations, we also show a wide range of pictorial representations of "the future with computers."




## Prophets of Computing

*Visions of Society Transformed by Computing*

**Dick van Lente**, **Editor**
*Erasmus University (retired)*

ISBN: 9781450398152
DOI: 10.1145/3548585

**http://books.acm.org**
**http://store.morganclaypool.com/acm**

DOI:10.1145/3547137

**Looking past inessential complexities to explain the Internet's simple yet daring design.**

BY JAMES MCCAULEY, SCOTT SHENKER, AND GEORGE VARGHESE

# Extracting the Essential Simplicity of the Internet

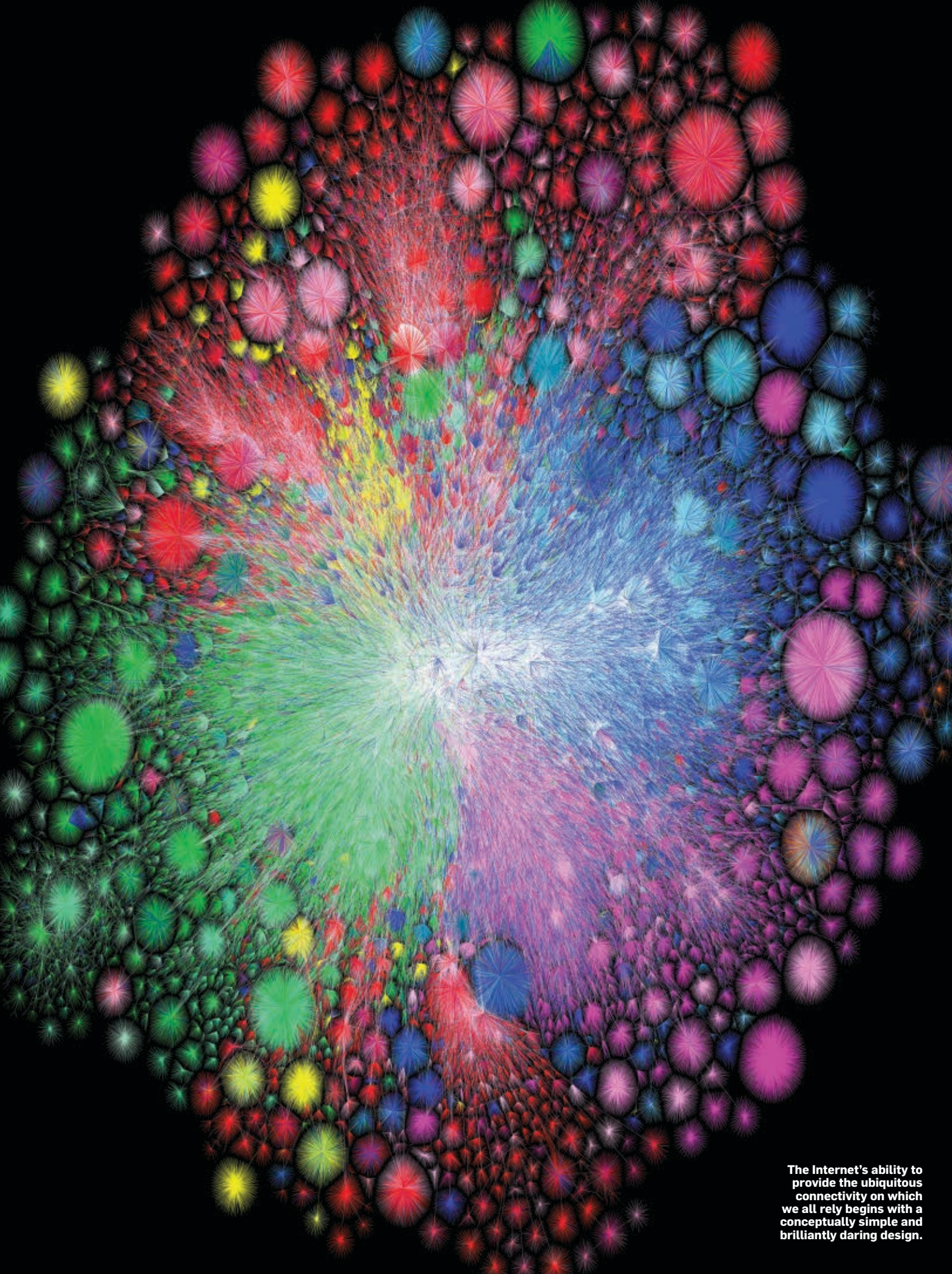
EVERYONE KNOWS THAT the Internet provides the ubiquitous connectivity on which we now rely, and many know that the underlying design was invented in the 1970s to allow computers to exchange data. However, far fewer understand the remarkable foresight that guided its design, which has remained essentially unchanged since its adoption in 1983 (see the sidebar "Brief Internet Timeline") while gracefully accommodating radical changes in applications, technologies, size, scope, and capacity. The Internet's design is underappreciated because its beauty is buried beneath an avalanche of implementation details. Most undergraduate networking courses teach students how the current Internet works, which requires they master an alphabet soup of protocols such as IP, TCP, BGP, OSPF, DVRP, DNS, STP, and ARP. These protocols are complex and riddled with details that are more historical than principled. As a result, much of the computer science community thinks that the Internet is merely a collection of complicated protocols, not a conceptually simple and brilliantly daring design.

This article attempts to extract the Internet's essential simplicity by motivating the fundamental Internet design choices from first principles without delving into all the ensuing complications. This allows us to focus on the *why* of the Internet rather than the *how*. Our first-principle analysis does not reflect the historical reasons for making these decisions (for such a discussion, see Cerf and Kahn[4] and Clark[6]) but instead justifies those decisions, as embodied in today's Internet (such as in the aforementioned protocols), with the full benefit of hindsight and from our own personal perspectives. While our purpose is primarily pedagogical, we also wrote this article as an homage to the pioneers[a] who shaped the Internet's design. Many of

a We do not have enough space to accurately and adequately credit the many people who helped create the Internet and guided its early evolution, but please see Leiner et al.[14] for more details on the who, what, and when of the Internet.

## » key insights

- **This article's thesis is that the Internet can best be understood as a small set of design choices: a service model, a four-layer architecture, and three crucial mechanisms (routing, reliability, and resolution).**
- **This article focuses on the rationale behind each of these design choices rather than on the details of the Internet's many protocols. While this "why, not how" approach is not sufficient to understand the complexities of today's Internet, it does enable readers to design a new Internet with approximately the same properties.**
- **This approach also illuminates the Internet's brilliant and daring design, which contains lessons for all computer scientists.**



**The Internet's ability to provide the ubiquitous connectivity on which we all rely begins with a conceptually simple and brilliantly daring design.**

## Brief Internet Timeline

**1969:** First exchange of data between remote computers. On October 29, 1969, a connection was established between UCLA and the Stanford Research Institute (SRI).

**1972:** Public demonstration of ARPANET. This was the first time the public was exposed to the promise of packet networks.

**1974:** Publication of the first paper describing the TCP/IP protocols.[4] These are the foundational protocols of today's Internet.

**1982:** DNS deployed, replacing the centralized distribution of a file that listed the IP address of every named host.

**1983:** ARPANET transitions to TCP/IP. On a global "flag-day" (January 1, 1983), all ARPANET systems finished converting to a four-layer architecture, which remains in use today.

**1985:** Spanning Tree Protocol (STP) invented, allowing Ethernets to safely interconnect by eliminating network loops.

**1989:** BGP design published. BGP is the interdomain routing protocol that today interconnects the Internet's many ASes.

**1990s:** The Internet becomes available to the public. The foundations of the Web were released in 1993, Yahoo was founded in 1994, and Akamai and Google were founded in 1998.

The Internet was established through the interconnection of several pioneering packet switched networks, key among them being ARPANET. Through a series of intermediate stages, including the development and integration of JANET in the U.K. and NSFNET in the U.S., these networks eventually became the public Internet. While the Internet infrastructure has changed greatly (in terms of speed, scope, and technologies), the basic technical design (in terms of its service model, architecture, and core protocol IP) has remained almost completely unchanged since 1983.

their basic design decisions were in direct conflict with the prevailing wisdom, arising from the telephone network, about how to build a global-scale communication infrastructure. Our attempt to extract simplicity in no way detracts from the brilliance of their achievement; on the contrary, we hope our presentation lets readers see that brilliance more clearly.

As background, we note that the Internet is a combination of network infrastructure (for example, links, switches, and routers) that delivers data between hosts, and host software (for example, applications along with networking support in libraries and operating systems) that processes that data. In this article, we start our search for simplicity by describing the delivery service supported by the network infrastructure, which we call the "service model." We then describe the Internet's "architecture," which is the modular decomposition of Internet functionality into four logical layers. This architecture requires three key mechanisms: **routing** (how to forward packets through the Internet so they reach their destination), **reliable delivery** (how to make data transfers resilient to packet losses), and **name resolution** (how to translate from names used by people and applications to addresses used by routing). After describing the designs of these three mechanisms, we then identify the key factors behind the Internet's success and end with a discussion the Internet's current weaknesses.

### Service Model

All host software must rely on the infrastructure's delivery service (that is, its service model) when communicating with other hosts. Thus, the design of a service model requires a compromise between what would best suit host software and what the infrastructure can feasibly support. In designing a data-transfer service model, we must choose its unit of transmission and its performance assurances. Given the bursty nature of computer communications, a small unit of transmission is necessary to achieve efficient resource utilization. The Internet uses a collection of bits called a packet, which today is typically no larger than 1.5kb.

Internet applications have a wide variety of network performance requirements, from low latency (interactive video or closed-loop control) to high bandwidth (transferring large datasets) and reliable delivery (file transfers). A natural engineering instinct would be to support the union of these requirements with a network infrastructure that guarantees specific bounds on latency, bandwidth, and reliability. However, the Internet, as its name connotes, must interconnect a wide variety of local packet networks—for example, wireless and wired, shared access, and point-to-point—many of which cannot guarantee performance. The lowest common denominator of what these packet networks can support is "best-effort" packet delivery, where the network attempts to deliver all packets but makes no assurances about if, when, or at what rate packets will be delivered.

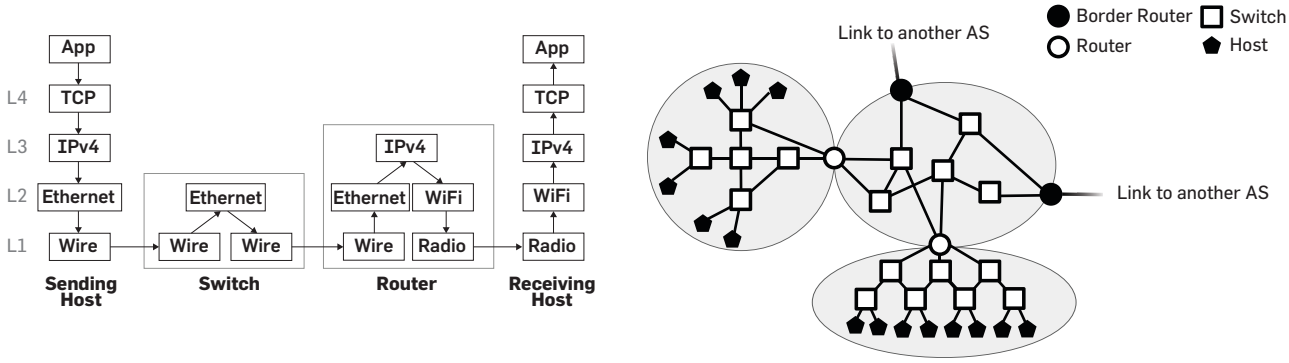
Thus, the fundamental question about the service model is whether it is better to be technologically ambitious and support all application requirements (but limit the possible packet networks that the Internet can incorporate) or accommodate all packet networks and provide only a modest, lowest-common-denominator, best-effort service model. As engineers, we naturally gravitate toward the intellectual challenge of ambitious technology goals. The Internet's designers instead opted for modesty, adopting the best-effort service model.

In hindsight, this was the superior choice for two reasons. First, the looser service model imposed minimal requirements on the infrastructure, enabling the Internet's extremely rapid growth. In short, being modest in performance requirements allowed the Internet to be ambitious in the speed and scope of its deployment. Second, the belief that applications have strict network requirements was a holdover from the telephone network, which had unintelligent end systems attached to a smart network that was fully responsible for meeting the strict performance requirements of telephony—that is, fixed-rate transmissions and reliable delivery. With increasingly powerful computers as the endpoints, Internet applications can be designed to adapt to various levels of performance, thereby loosening their requirements. Network operators can also ensure reasonable performance in most circumstances by providing sufficient bandwidth—that is, by deploying more and faster links so that the network is rarely congested.

### Architecture

Architecture is about the organization of functionality not its implementation. Modularity is a guiding principle for architecture, calling for the decomposition of a system's goal into

**Left: A packet's path through the layers in hosts, routers, and switches. Right: An AS with switches forwarding at L2 within networks (denoted by shaded ovals), routers forwarding between networks at L3, and border routers connecting to other ASes.**




smaller tasks with clean interfaces. The Internet's goal—allowing applications on two different computers to communicate—can first be decomposed into two components: (i) what the network infrastructure does (best-effort packet delivery between hosts) and (ii) what the networking support software on the host does (making it easier for applications to use this best-effort delivery). We now discuss these two components and address four additional questions.

**Network infrastructure.** The infrastructure's job of host-to-host delivery can be decomposed into three different tasks arranged in layers, with the higher layers having broader geographic scope and the lower layers handling more local tasks. The process of delivering a packet from sending host to receiving host thus involves stitching together a set of these local delivery tasks. The most local and low-level task in providing best-effort packet delivery is the transmission of bits over links or broadcast media, which requires digital-to-analog conversion and error-correction, neither of which are specific to the Internet. This local-delivery task is handled by what is called the Physical Layer (L1).

Given L1's ability to send bits over links, the next task is to enable communication within a local packet network (hereafter, the term "network" will refer to locally scoped designs such as an Ethernet or a wireless network). This involves two steps: (i) forming packets out of sets of bits prepended with a packet header that describes where those bits should be delivered, and then (ii) delivering those packets to the appropriate destination within the local network. Limiting this second step to local (not global) delivery enables the use of non-scalable techniques such as *broadcast*, where the transmission medium itself ensures broadcast packets reach all hosts (as in wireless), and *flooding*, where the network ensures that flooded packets reach all hosts. This task is handled by what we call the Network Layer (traditionally called the Link Layer, but that terminology is no longer appropriate) or L2. In non-broadcast networks, this task is implemented by switches within the local network that forward packets to their destination.

The last task is to carry packets from the sender's network to the destination's network, leveraging L2's ability to deliver packets within these networks to individual hosts. This interconnection of networks is handled by the Internetworking Layer (L3) and is implemented by routers that are connected to two or more networks at L2 (by connected at L2, we mean that each of these networks can be reached from the router without going through another router). Packets are forwarded through a series of routers (with router-to-router delivery supported by L2 within the network to which both routers are connected) until they reach the destination network. This layer interconnects the networks, which is memorialized in the term "Internet." It also defines the Internet's service model, in that L3 delivers data to the higher-layer host software. See the figure above for a depiction of how a packet travels through these layers as it traverses the Internet.

**Network support in host software.** The network infrastructure's host-to-host service model does not specify to which host application the packets should be delivered, and its best-effort and packet-oriented nature can be hard for applications to use without additional help. To rectify these problems, host operating systems offer the Transport Layer (L4) in addition to other networking support. L4 delivers packets to individual applications on the host based on metadata in the packet header called a "port." To make best-effort service easier for applications to use, some common transport protocols offer three functions. The first is a byte-stream API so applications can write and read data using a simple file-like interface rather than having to explicitly send and receive individual packets. The second is controlling a host's rate of packet transmissions to prevent network overloads; this is called congestion control and involves a control loop, where transport protocols reduce their sending rate if they detect network congestion—for example, via dropped packets or increased delay. There are many congestion-control algorithms, differing in how they detect and respond to congestion, but the use of such control loops is conceptually (not practically) simple, so we do not delve into their details. The third, and most fundamental, is reliable packet delivery, so applications need not cope with packet losses. Such losses are a natural part of the network infrastructure's best-effort service model but are unacceptable to many applications. Although reliability, for the applications that require it, can be implemented in applications themselves or in other supporting li-

braries, for the sake of clarity, we focus on the very common and traditional arrangement, where reliability is implemented in L4.

**Four additional questions.** The four-layer Internet architecture—Physical Layer (L1), Network Layer (L2), Internetworking Layer (L3), and Transport Layer (L4)—is the natural result of modularity. Each layer only interacts with the layers directly above and below. Note that since packets always arrive via physical media at L1 (as shown in the figure), hosts must implement all four layers; routers implement the first three while switches only implement the first two. To complete our discussion, we must address four questions.

*Question #1: What kind of diversity does this layered architecture enable and how is it managed?* As long as two implementations of a layer offer the same upward and downward interfaces, they are architecturally interchangeable. In addition, the Internet architecture's modularity allows the coexistence of multiple transport protocols at L4, multiple local network designs at L2, and multiple physical technologies at L1, each offering their own distinct interfaces. The L1 and L2 technologies are chosen by the network provider, who can ensure that their interfaces are compatible—that is, that the local network design can use the set of link technologies. An application chooses its desired L4 protocol when it invokes the operating system's network API. Applications and network providers make their choices independently—that is, an application does not know about the network in which it currently resides, and the network provider does not know which applications will be used on its network. This works seamlessly if and only if there is a single set of interfaces at L3 that all L2 designs and L4 protocols are compatible with. Thus, we must have a single protocol at L3, which is the Internet Protocol (IP), currently IPv4. The next version, IPv6, is gaining popularity and has essentially the same interface as IPv4. IP functions as the "narrow waist" of the Internet architecture, and its singularity enables plurality (and thus innovation) at all other layers.

*Question #2: What identifiers does the Internet use?* The Internet must enable L2 and L3 packet headers to identify destinations for routing and allow users and applications to identify the services they want to access. This results in three kinds of names or addresses, which we now describe in turn. Typically, each Internet-reachable hardware network interface on a host (such as Wi-Fi, cellular, or Ethernet cards) has a permanent and unique address called a MAC address, which is used by L2 to identify destinations. L3 uses IP addresses, which specify a unique network within the Internet, and a specific host interface currently using the address within that network (such address assignments can change over time, but for clarity, we do not consider that detail here). Users and applications employ application-level names to refer to host-based services. To give some degree of permanence to such names, they are independent of whichever machine(s) the service is based on as well as where those hosts might be placed in the network. Of these three identifiers—application-level names, IP addresses, and MAC addresses—the first two must be resolved to the identifier at the next-lowest level. Thus, when an application on one host attempts to send a packet to an application on another host, it must resolve the application-level name into an IP address. When a packet arrives at a network, it is sent via L2 to the destination host or (as we explain below) to the next-hop router. In both cases, an IP address must be resolved into a MAC address. We describe the resolution of application-level names and the resolution of IP addresses later in this article.

*Question #3: How is the Internet infrastructure organized?* The Internet is not just an unstructured set of L2 networks interconnected by routers. Instead, it is a collection of Autonomous Systems (ASes), also called domains, each of which contains a set of L2 networks managed by a single entity. Examples of ASes include enterprises, universities, and commercial Internet Service Providers (ISPs). These ASes control their own internal L3 routing and must engage in a routing protocol with other ASes to provide host-to-host delivery between ASes. Thus, L3 involves two routing tasks: (i) routing between networks *within* an AS (intradomain routing), which is handled by routers in that AS, and (ii) routing *between* ASes (interdomain routing), which is handled by so-called border routers that connect two or more ASes. As we will discuss, these two routing tasks have different requirements and thus require different routing paradigms and protocols.

*Question #4: How do all these pieces fit together?* The process of delivering a packet across the Internet starts with an application resolving an application-level name to an IP address and then invoking a host's networking support to send data to that destination IP address. This results in a call to L4, which packetizes the data and invokes L3 to deliver those packets. At L3, a packet is forwarded through a sequence of routers until it reaches its destination network (as identified by the destination IP address in the packet header). Each router has a forwarding table that maps a destination IP address into the IP address of the next-hop router. Upon receipt of a packet, a router looks up the appropriate next-hop router in the forwarding table and then sends the packet to that next-hop node by calling L2. L2 must first resolve the IP address of the next-hop router into a MAC address and then deliver the packet to the next-hop router (either by broadcast or by forwarding the packet through a series of switches as explained in the next section), which then returns the packet to L3. The key technical challenge in this process is setting up the L3 forwarding tables so that the set of next-hops always results in the packet arriving at the appropriate destination.

Our discussion to this point has identified three nontrivial mechanisms—routing, reliability, and resolution—that are necessary to implement the Internet architecture. The following three sections discuss these in turn.

## Routing

In this section, we use the term "routing" to refer to the generic problem of forwarding packets through the Internet to an intended destination host, whether that occurs at L3 and is implemented by routers, or at L2, where it is implemented by switches and thus is often called switching rather than routing. We start by considering L3 intradomain routing, where we assume: (i) each L3 packet header contains a destination IP address, (ii) each router has a

set of neighboring routers to which it is connected at L2, and (iii) each router has a forwarding table that correctly indicates whether the router is connected (at L2) to the packet's destination network and, if not, deterministically maps the destination of an arriving packet to the neighboring next-hop router the packet should be forwarded to. Before discussing how the forwarding table is established, we focus on the conditions under which a static set of forwarding tables successfully guides packets toward their destination. We refer to the set of interconnections between routers as the network graph, which we assume is connected; the set of forwarding tables as the forwarding state; and the union of these as the routing state. We say that a given instance of the routing state is *valid* if it always directs packets to their destination; we say it has a *loop* if there are cases (that is, a starting position and a destination address) where a packet can return to a router it already visited.

**Result:** *An instance of routing state is valid if and only if there are no loops.*

**Argument:** *Assume there are no loops; because the network is connected and finite, any packet must eventually hit a router that is connected (at L2) to its destination. Assume there is a loop for a given destination; any packet addressed to that destination which enters the loop will never reach the destination.*

Why do we care about this obvious result? Because the Internet uses a variety of routing algorithms to compute the forwarding state, and the key conceptual difference between these routing algorithms lies in the way they avoid loops in steady state. Routing protocols are complicated distributed systems that solve the problem of how to recompute the forwarding state quickly and automatically as the network graph changes due to network links failing and recovering. Here, we avoid the complexities of these distributed protocols and only focus on the forwarding state that results when the algorithm has converged to a steady state on a static network. The methods for avoiding loops in steady state depend crucially on what information the routers share with each other in these algorithms.

For example, a router knows its neighboring routers and can use a flooding algorithm to share this local information with every other router. In steady state, every router could use this information to assemble the entire network graph. If all routers use the same loop-free path-finding algorithm on this shared network graph to compute their forwarding tables, the resulting routing state is always valid. This approach is called "link-state" routing.

**Much of the computer science community thinks that the Internet is merely a collection of complicated protocols, not a conceptually simple and brilliantly daring design.**

Another case is when each router informs its neighboring routers of its own distance to all other networks, according to some metric, such as latency. Router $\alpha$ can compute its distance to each destination network $n$ as $d_\alpha(n) = \min_\beta [d_\beta(n) + d(\alpha, \beta)]$, where $d_\beta(n)$ is the distance from each neighboring router $\beta$ to network $n$, and $d(\alpha, \beta)$ is the distance between the two routers $\alpha, \beta$. The steady state of this distributed computation produces shortest-path routes to each destination, which cannot have loops. This is called "distance-vector" routing.

When there is a change in the network graph, temporary loops in the routing state can arise during the process of recomputing routes (that is, when the protocol has not yet converged to the steady state) in both distance-vector and link-state routing. If left unchecked, packets cycling endlessly around such loops could cause severe network congestion. To prevent this, the IP protocol wisely incorporated a field in the packet header that starts with an initial number set by the sending host and which is then decremented every time a packet reaches a new router. If this field reaches zero, the packet is dropped, limiting the number of times packets can traverse an ephemeral loop and thereby ensuring that ephemeral loops are not catastrophic. Without this simple mechanism, all routing protocols would need to preclude ephemeral loops, and this degree of care would likely complicate the routing protocol.

We next consider L3 interdomain routing. ASes clearly must carry all packets for which they are the source or the destination; all other packets are considered *transit* traffic, passing through an AS on the way from the source AS to the destination AS. ASes want the freedom to choose both what transit traffic they carry and which

routes they use to forward their traffic toward its destination. For ISPs, these two policy choices depend heavily on their economic arrangements with their neighboring ASes, so they want to keep these choices private to avoid revealing information that would help their competitors.

While interdomain routing choices are in practice implemented by border routers, and this involves some complicated intradomain coordination between them, we can simplistically model interdomain routing with the collection of interconnected ASes forming a graph and the routing decisions being made by the ASes themselves. Interdomain routing must: (i) enable ASes to make arbitrary policy choices, so we cannot use distance-vector, and (ii) keep these policy choices private, so we cannot use link-state routing, which would require ASes to make their policies explicit so that every other AS could compute routes using those policies. An alternate approach is to allow ASes to implement their policies by choosing to whom they advertise their routes (by sending a message to a neighboring AS saying, "You can use my path to this destination") and choosing which routes they use when several of their neighbors have sent them routes to a given destination. These are the same messages and choices used in distance-vector routing, but distance-vector allows no policy flexibility: Routes are advertised to all neighbors, and only the shortest paths are chosen. This local freedom provides policy flexibility and privacy, but how do we prevent steady-state loops when routes are calculated, given that ASes can make arbitrary and independent choices about which routes they select and to which neighbors they offer their routes? The Internet's solution is to exchange path information. When AS "A" advertises a route (for a particular destination) to neighboring AS "B", it specifies the entire AS-level path of that traffic to the destination. If AS "B" sees it is already on that path, it does not use that route, as it would create a loop. If all ASes obey this obvious constraint, the steady state of what we call "path-vector" routing will be loop-free regardless of what policy choices ASes make. Path-vector routing is used in BGP, which is the current interdomain routing protocol; as such, BGP is the glue that holds the Internet's many ASes together.

**Being modest in performance requirements allowed the Internet to be ambitious in the speed and scope of its deployment.**

This path-vector approach ensures that there are no loops in any steady state. However, it does not ensure that the routing protocol converges to a steady state—for example, the policy oscillations first described in Varadhan et al.[17] are cases where path-vector algorithms do not converge. Nor does it ensure that all resulting steady states provide connectivity between all endpoints—for example, all ASes could refuse to provide transit connectivity to a given AS. Researchers were confused as to why these anomalies were not observed in the Internet, but the theoretical analysis in Gao and Rexford[9] (and subsequent work) showed that typical operational practices (where there is a natural hierarchy of service providers, lower-tier providers are customers of higher-tier providers, top-tier providers are connected in a full mesh, and routes are chosen to maximize revenue and minimize costs) produce routing policies that will always converge to steady states that provide end-to-end connectivity between all endpoints.

Avoiding loops also plays a role at L2 in non-broadcast networks, where flooding is often used to reach a destination. The spanning-tree protocol (STP) creates a tree out of the network (that is, eliminates all cycles from the network graph) by choosing to not use certain links. Once the network is a spanning tree, one can flood a packet to all hosts by having each switch forward the packet on all neighboring links that are part of the spanning tree, aside from the one by which the packet arrived. Such flooding allows hosts and routers to resolve IP addresses into MAC addresses by flooding an "address resolution protocol" (ARP) message that asks, "Which host or router has this IP address?"; the owner then responds with its MAC address. During this ARP exchange (and in fact whenever a host sends a packet), switches can learn how to reach a particular host without flooding by remembering the link on which they have most recently received a packet from that host. There is only one path between any two nodes on a spanning tree, so a host can be reached by sending packets over the link on which packets from that host

arrived. Thus, with such "learning switches," the act of resolving an IP address into a MAC address lays down the forwarding state between the sending and receiving hosts. When sending packets to a host where the MAC address has already been resolved, the network need not use flooding but can deliver the packets directly.

To summarize, all routing algorithms must have a mechanism to avoid loops in steady state (that is, after the protocol has converged), which in turn depends on what information is exchanged. To limit information exchanges to neighboring routers within ASes, one should use distance-vector, which results in shortest-paths to guarantee loop-freeness. To improve route flexibility in intradomain routing, a better choice is link-state, which requires flooding of neighbor information but allows an arbitrary, loop-free path calculation to be used. To allow ASes to exercise individual policy control in interdomain routing, they can exchange explicit path information to avoid loops. To enable flooding and learning routes on the fly in L2, it is useful to turn network graphs into spanning trees as they are inherently loop-free. There are other issues one might consider in routing, such as how to recover from failures without having to recompute routes (as in the use of failover routes[5]) and how one might use centralized control to simplify routing protocols (as in SDN[18]), but here our focus is on articulating the role of loop avoidance in commonly used routing paradigms.

## Reliable Delivery

Our discussion of routing analyzed when routing state would enable the delivery of packets, but even with valid routing state, packets can be dropped due to overloaded links or malfunctioning routers. The Internet architecture does not guarantee reliability in the first three layers, but instead wisely (as argued in Saltzer et al.[15]) leaves this to the transport layer or applications themselves; dropped packets will only be delivered if they are retransmitted by the sending host.

Here we consider reliable delivery as ensured by a transport protocol that establishes connections which take data from one application and transmit it to a remote application. Several important transport protocols, such as the widely used TCP, provide the abstraction of a reliable byte-stream, where data in the byte-stream is broken into packets and delivered in order, and all packet losses are recovered by the transport protocol itself. The reliable byte-stream abstraction can be implemented entirely by host software that can distinguish packets in one byte-stream from packets in another, has sequence numbers on packets so they can be properly reordered before any data is handed to the receiving application, and retransmits packets until they have been successfully delivered.

Informally, a reliable transport protocol takes data from an application, transmits it in packets to the destination, and eventually either informs the application that the transfer has successfully completed or that it has terminally failed—and in both cases ceases further transmissions. For our discussion, we eliminate the possibility of announcing failure, since every transport protocol that eventually announces failure is by definition reliable. However, we assume that the underlying network eventually delivers a packet that has been repeatedly sent, so a persistent protocol can always succeed (more precisely, we assume that a packet that has been retransmitted infinitely often is delivered infinitely often). For this case, we ask: what communication is needed between the sender and receiver to ensure that the protocol can inform the application that it has succeeded if and only if all packets have been received? There are two common approaches: the receiver can send an acknowledgement (ACK) to the sender whenever it receives a packet, or it can send a non-acknowledgement (NACK) to the sender whenever it suspects that a packet has been lost.

**Result:** *ACKs are necessary and sufficient for reliable transport, while NACKs are neither necessary nor sufficient.*

**Argument:** *A reliable transport protocol can declare success only when it knows all packets have been delivered, and that can only be surmised by the receipt of an ACK for each of them. The absence of a NACK (which itself can be dropped) does not mean that a packet has been received.*

Note that NACKs can be useful, as they may provide more timely information about when the sender should retransmit. For example, TCP uses explicit ACKs for reliability and initiates retransmissions based on timeouts and implicit NACKs (when the expected ACK does not arrive).

## Name Resolution

In addition to resolving IP addresses into MAC addresses via ARP, the Internet must also resolve application-level names into one or more IP addresses. These names are informally called hostnames and formally called fully qualified domain names. We use the nonstandard terminology application-level names because they neither refer to a specific physical machine (whereas MAC addresses do) nor do they directly relate to the notion of domains used in interdomain routing.

Any application-level naming system must: (i) assign administrative control over each name to a unique authority that decides which IP address(es) the name resolves to; (ii) handle a high rate of resolution requests; and (iii) provide both properties at a scale of billions of application-level names. To meet these challenges, the Internet adopted a hierarchical naming structure called the Domain Name System (DNS). The namespace is broken into regions called domains, which are recursively subdivided into smaller domains, and both resolution and administrative control are done hierarchically. Each naming domain has one or more nameservers that can create new subdomains and resolve names within its domain. Names can either be "fully qualified" (these resolve to one or more IP addresses) or not (their resolution points to one or more subdomain nameservers that can further resolve such names). This hierarchy begins with a set of top-level domains (TLDs) such as .cn, .fr, .ng, .br, .com, and .edu, which are controlled by a global authority (ICANN). Commercial registries allow customers to register subdomains under these TLDs. The resolution of TLDs to their nameservers is handled by a set of DNS root servers (whose addresses are known by all hosts), and the resolution proceeds

down the naming hierarchy from there. For example, www.myschool.edu is resolved first by a root server that points to a nameserver for .edu; the nameserver for .edu points to a nameserver for myschool.edu, which then resolves www.myschool.edu into an IP address. This hierarchical structure allows for highly parallel name resolution and fully decentralized administrative control, both of which are crucial for handling the scale of Internet naming.

## Secrets to the Internet's Success

We have tried to reduce the Internet's impenetrable complexity into a small set of design choices: (i) a service model, (ii) a four-layer architecture, and (iii) three crucial mechanisms (routing, reliability, and resolution). Understanding the reasoning behind these decisions is not sufficient to understand the complexities of today's Internet but is sufficient to design a new Internet with roughly the same properties. This is consistent with our goal, which is to extract the Internet's essential simplicity. Unfortunately, this simplicity is not sufficient to explain the Internet's longevity, so we now ask: *Why has the Internet's design been so successful in coping with massive changes in speed, scale, scope, technologies, and usage?* We think there are four key reasons:

**Modesty.** Rather than attempt to meet all possible application requirements, the Internet adopted a very modest but quite general service model that makes no guarantees. This gamble on smart hosts and unintelligent networks: (i) allowed new applications to flourish, because the Internet had not been tailored to any specific application requirements; (ii) leveraged the ability of hosts to adapt in various ways to the vagaries of best-effort Internet service (for example, by rate adaptation, buffering, and retransmission); and (iii) enabled a rapid increase in network speeds, since the service model was relatively simple. If the Internet had adopted a more complicated service model, it might have limited itself to the requirements of applications that were present at the time of creation and employed designs that could have been implemented with then-available technologies. This would have led to a complicated design tailored to a narrow class of applications and quickly outmoded technologies, a recipe for short-term success but long-term failure.

**Modularity.** The modularity of the four-layer Internet architecture resulted in a clean division of responsibility: The network infrastructure (L1/L2/L3) supports better packet delivery (in terms of capacity, reach, and resilience) while applications (assisted by L4) create new functionality for users based on this packet-delivery service model. Thus, the architecture allowed two distinct ecosystems—network infrastructure and Internet applications—to flourish independently. However, the Internet's modularity goes beyond its formal architecture to a more general approach of maximizing autonomy within the confines of its standards-driven infrastructure, which is in sharp contrast to the more rigid uniformity of the telephone network. For instance, the only requirement on an AS is that its routers support IP and participate in the interdomain routing protocol BGP. Otherwise, each AS can deploy any L1 and L2 technologies and any intradomain routing protocol without coordinating with other ASes. Similarly, individual naming domains must support the DNS protocol but otherwise can adopt whatever name management policies and name resolution infrastructures they choose. This infrastructural autonomy allowed different operational practices to arise. For example, a university campus network, a hyperscaled datacenter network, and an ISP backbone network have very different operational requirements; the Internet's inherent autonomy allows them to meet their needs in their own way and evolve such methods over time.

**Assuming failure is the normal case.** As systems scale in size, it becomes increasingly likely that, at any point in time, some components of the system have failed.[10] Thus, while we typically think of scaling in terms of algorithmic complexity or state explosion, handling failures efficiently is also a key scalability requirement. In contrast to systems that expect normal operation and go into special modes to recover from failure, almost all Internet mechanisms treat failure as a common occurrence. For instance, in basic routing algorithms, the calculation of routes is typically done in the same way whether it is due to a link failure or a periodic recalculation. Similarly, when a packet is lost, it must be retransmitted, but such retransmissions are expected to occur frequently and are not a special case in transport protocols. This design style—of treating failures as the common case—was the foundation of Google's approach to building its hyperscaled infrastructure,[2] and we should remember that it was first pioneered in the Internet.

**Rough consensus and running code.** While this article has focused on the Internet's technical aspects, the process by which it was designed was also key to its success. Rather than adopt the formal design committees then current in the telephony world, the Internet inventors explicitly chose another path: Encourage smaller groups to build designs that work, and then choose, as a community, which designs to adopt. This was immortalized in a talk given by David Clark, one of the Internet's architectural leaders, who said, "We reject kings, presidents, and voting. We believe in rough consensus and running code." This egalitarian spirit extended to the shared vision of the Internet as a uniform and unifying communication platform connecting all users. Thus, the development of the Internet was indelibly shaped not only by purely technical decisions but also by the early Internet community's fervent belief in the value of a common platform connecting the world and their communal ownership of the design that would realize that vision.

## Nothing Is Perfect

There are many areas where the Internet's design is suboptimal,[6,7] but most of them are low-level details that would not change our high-level presentation. In this section, we review three areas where the Internet's design has more fundamental problems.

**Security.** Many blame the poor state of Internet security on the fact that security was not a primary consideration in its design, though survivability in the face of failure was indeed an important consideration.[6] We think this critique is misguided for two reasons: (i) security in an interconnected world is a far more complicated and elusive goal than network security, and the Internet can only ensure the latter, and (ii) while

the Internet architecture does not inherently provide network security, there are protocols and techniques, some of which are in wide use, that can largely achieve this goal.

To be more precise, we say (similar to Ghodsi et al.[11]) that a network is secure if it can ensure the following properties when transferring data between two hosts: (i) connectivity between hosts is reasonably reliable (*availability*); (ii) the receiver can tell from which source the data came (*provenance*); (iii) the data has not been tampered with in transit (*integrity*); and (iv) the data has not been read by any intermediaries and no one snooping on a link can tell which hosts are exchanging packets (*privacy*). The latter three can be ensured by cryptographic protocols. The Internet's availability is vulnerable to distributed denial-of-service (DDoS) attacks, where many hosts (typically compromised hosts called bots) overload the target with traffic. There are techniques for filtering out this attacking traffic, but they are only partially effective. Availability is also threatened by BGP's vulnerability to attacks where ASes lie about their routes; cryptographic solutions are available, but they are not widely deployed. Thus, there are cryptographic protocols and mitigation methods that would make the Internet largely satisfy the definition of a secure network.

However, the failure to make the Internet an inherently secure network is most definitely not the primary cause of our current security issues, because a secure network does not prevent host software from being insecure. For example, if an application is designed to leak information about its users, or impersonate its users to execute unwanted financial trades, there is little the network can do to prevent such malicious actions. A host can be infected by a malicious application if it is unwittingly downloaded by a user or if some attack (such as a buffer overflow) turns a benign application into a malicious one. Comments about the Internet's woeful security typically refer to the ease of placing malicious software on hosts, but this is not primarily a network security problem.

**In-network packet-processing.** The early Internet designers argued[15] that the network infrastructure should generally focus on packet delivery and

**Rather than adopt the formal design committees then current in the telephony world, the Internet inventors explicitly chose another path.**

avoid higher-layer functionality, but this precept against what we call in-network packet-processing is currently violated in almost every operational network (see Blumenthal and Clark[3]). Modern networks have many middleboxes that provide more-than-delivery functions such as firewalling (controlling which packets can enter a network), WAN optimization (increasing the efficiency of networks by caching previous transfers), and load balancing (directing requests to underloaded hosts). A 2012 survey[16] revealed that roughly one-third of network elements are middleboxes, one-third are routers, and one-third are switches. Some of these in-network processing functions are deployed within a single enterprise to improve their internal efficiency; such violations only have impact within an enterprise network so are now widely viewed as an acceptable fact of life. In addition, as discussed below, several cloud and content providers (CCPs) have deployed large private networks that deploy in-network functions—particularly flow termination, caching, and load balancing—that lower the latency and increase the reliability seen by their clients.

**Lack of evolvability.** Since IP is implemented in every router, it is very difficult to change the Internet's service model. Even the transition to IPv6, which began in 1998 and has accelerated recently due to IPv4 address shortages, retains the same best-effort service model. The Internet has remained without significant architectural change for decades because its service has been "good enough"[12] and there was no viable alternative to the Internet. However, the large private networks deployed by the CCPs now offer superior service to their clients via their in-network processing, and most client traffic today either is served within its originating AS by a local CCP cache or goes directly from the source AS to a CCP's large private network with its specialized in-network processing.[1,18,13,19] The CCPs' large private networks are vertically integrated with cloud and content services that are now more dominant economic forces than ISPs. With the Internet unable to evolve to adopt similar functionality, the growth of

these private networks is likely to continue.

Thus, we might already be seeing the beginning of a transition to a new global infrastructure in which these private networks have replaced all but the last mile of the current Internet (that is, the access lines to residences and businesses) for almost all traffic. This would mark the end of the Internet's age of economic innocence, when it had no serious competitors and could settle for "good enough." While the Internet faces many challenges, we think that none are as important to its future as resolving the dichotomy of visions between the one that animated the early Internet community, which is that of a uniform Internet connecting all users and largely divorced from services offered at endpoints, and the one represented by the emerging large private CCP networks, in which the use of in-network processing provides better performance but only to clients of their cloud or content services.

Despite this gloomy prospect, we wrote this article to praise the Internet not bury it. The Internet is an engineering miracle, embodying design decisions that were remarkably prescient and daring. We should not let the complexities of today's artifact obscure the simplicity and brilliance of its core design, which contains lessons for us all. And we must not forget the intellectual courage, community spirit, and noble vision that led to its creation, which may be the Internet's most powerful lesson of all.


## Acknowledgments

We thank our anonymous reviewers as well as our early readers—including Vint Cerf, David Clark, Deborah Estrin, Ethan Katz-Bassett, Arvind Krishnamurthy, Jennifer Rexford, Ion Stoica, and a dozen Berkeley graduate students who read the paper under duress—for their many helpful comments, but all remaining errors are entirely our responsibility. C


**The Internet was indelibly shaped not only by purely technical decisions, but also by a fervent belief in the value of a common platform connecting the world.**


**References**

1. Arnold, T. et al. Cloud provider connectivity in the flat Internet. In *Proceedings of the ACM Internet Measurement Conference*, Association for Computing Machinery (2020), 230–246; https://doi.org/10.1145/3419394.3423613.
2. Barroso, L.A. and Hoelzle, U. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines* (1st ed.), Morgan and Claypool Publishers (2009).
3. Blumenthal, M.S. and Clark, D.D. Rethinking the design of the Internet: The end-to-end arguments vs. the brave new world. *ACM Transactions on Internet Technology 1*, 1 (August 2001), 70–109; https://doi.org/10.1145/383034.383037.
4. Cerf, V.G. and Kahn, R.E. A protocol for packet network intercommunication. *IEEE Transactions on Communications 22*, 5 (1974), 637–648; https://doi.org/10.1109/TCOM.1974.1092259.
5. Chiesa, M. et al. A survey of fast-recovery mechanisms in packet-switched networks. *IEEE Communications Surveys Tutorials 23*, 2 (2021), 1253–1301; https://doi.org/10.1109/COMST.2021.3063980.
6. Clark, D. The design philosophy of the DARPA Internet protocols. In *ACM SIGCOMM Computer Communication Review 18*, 4 (August 1988), 106–114; https://doi.org/10.1145/52324.52336.
7. Clark, D.D. *Designing an Internet* (1st ed.), The MIT Press (2018).
8. Labovitz, C. Pandemic impact on global Internet traffic. North American Network Operators Group (2019); http://bit.ly/3YCXy5c.
9. Gao, L. and Rexford, J. Stable Internet routing without global coordination. *IEEE/ACM Transactions on Networking 9*, 6 (2001), 681–692; https://doi.org/10.1109/90.974523.
10. Ghemawat, S., Gobioff, H., and Leung, S-T. The Google file system. In *Proceedings of the 19th ACM Symp. on Operating Systems Principles*, Association for Computing Machinery (2003), 29–43; https://doi.org/10.1145/945445.945450.
11. Ghodsi, A. et al. Naming in content-oriented architectures. In *Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking*, Association for Computing Machinery, (2011), 1–6; https://doi.org/10.1145/2018584.2018586.
12. Handley, M. Why the Internet only just works. *BT Technology Journal 24* (2006), 119–129.
13. Huston, G. The death of transit? *APNIC.net* (2016); https://blog.apnic.net/2016/10/28/the-death-of-transit/.
14. Leiner, B.M. et al. A brief history of the Internet. *Computer Communication Review 39*, 5 (2009), 22–31; https://doi.org/10.1145/1629607.1629613
15. Saltzer, J.H., Reed, D.P., and Clark, D.D. End-to-end arguments in system design. *ACM Transactions on Computer Systems 2*, 4 (November 1984), 277–288; https://doi.org/10.1145/357401.357402.
16. Sherry, J. et al. Making middleboxes someone else's problem: Network processing as a cloud service. *ACM SIGCOMM 2012 Conf.*, L. Eggert, J. Ott, V.N. Padmanabhan, and G. Varghese (Eds.), 13–24; https://doi.org/10.1145/2342356.2342359.
17. Varadhan, K., Govindan, R., and Estrin, D. Persistent route oscillations in inter-domain routing. *Computer Networks 32*, 1 (2000), 1–16; http://dblp.uni-trier.de/db/journals/cn/cn32.html#VaradhanGE00.
18. Wikipedia contributors. Software defined networking; https://en.wikipedia.org/wiki/Software-defined_networking.
19. Wohlfart, F. et al. Leveraging interconnections for performance: The serving infrastructure of a large CDN. In *Proceedings of the 2018 Conf. of the ACM Special Interest Group on Data Communication*. Association for Computing Machinery, 206–220; https://doi.org/10.1145/3230543.3230576.



**James McCauley** is an assistant professor at Mount Holyoke College, Computer Science, South Hadley, Massachusetts, USA.

**Scott Shenker** (shenker@icsi.berkeley.edu) is a professor at the University of California, Berkeley, EECS and a researcher at the International Computer Science Institute, Berkeley, California, USA.

**George Varghese** is the Jonathan B. Postel Professor of Computer Science at the University of California Los Angeles, USA.






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

DOI:10.1145/3554976

**Prior pessimism about reuse in software engineering research may have been a result of using the wrong methods to measure the wrong things.**

BY MARIA TERESA BALDASSARRE, NEIL ERNST, BEN HERMANN, TIM MENZIES, AND RAHUL YEDIDA

# (Re)Use of Research Results (Is Rampant)

ACCORDING TO POPPER,[23] the ideas we can most trust are those that have been the most tried and tested. For that reason, many of us are involved in this process called "science," which produces trusted knowledge by sharing one's ideas and trying out and testing the ideas of others. Science and scientists form communities where people do each other the courtesy of curating, clarifying, critiquing, and improving a large pool of ideas.

According to this definition, one measure of a scientific community's health is how much it reuses results. By that measure, the software engineering research community might seem to be very unhealthy. Da Silva et al. reported that from 1994 to 2010, only 72 studies had been replicated by 96 new studies.[10] In February 2022, as a double-check for da Silva's conclusion, we queried the ACM Portal for products from the International Conference on Software Engineering (ICSE), that field's premier conference. Between 2011 and 2021,

## » key insights

- **Researchers must take back control over how our products are shared.**
- **Results should not be locked away behind paywalls that block access to results and enshrine outdated views on science. Using open source tools, research communities can map the real patterns of inference between their work.**
- **For example, in SE, researchers often share many artifacts, ranging from ideas to paper-based methods, datasets, and tools. While exact replication of results is rare, we can report just from one year that there are hundreds of cases where one researcher used some but not all of the artifacts from other work.**
- **We ask others to join us in this effort to accurately record the reality of 21st century science.**

**Figure 1. The 1,635 arrows in this diagram connect reusers to the reused.**

Blue dots denote the 714 sources found with a digital object identifier (DOI)—for example, any paper from a peer-reviewed source. Red dots denote the 48 papers we found without DOIs—for example, those from arxiv.org. Gray and green dots denote the 297 works in grey literature and 57 GitHub repositories (respectively) reused in this sample. The black squares pull out four examples where a paper has reused material from dozens of other sources. Data collection for this graph is ongoing and, at the time of this writing, comes from 40% of the papers published in the 2020 technical program of ICSE, ASE, FSE, ICMSE, MSR, and ESEM. From the website https://reuse-dept.org.

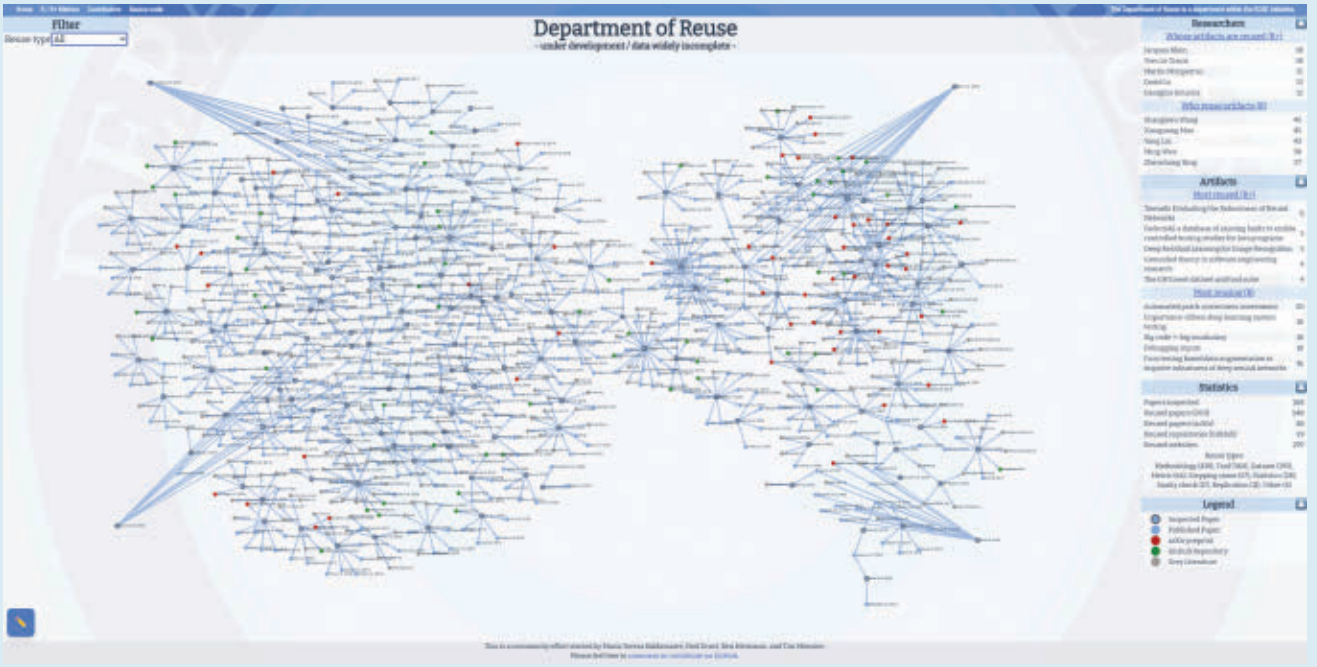



**Table 1. Badges such as the ones shown in this table are currently awarded at conferences.[2] This table is based on ACM's badge program, however, analogous badges are used at other conferences. Images used by permission of the Association for Computing Machinery.**

| Available | Functional | Reusable | Reproduced | Replicated |
|---|---|---|---|---|
|  |  |  |  |  |
| In a public repository with a long-term retention policy. A DOI needs to be provided. | Artifacts are documented, consistent, complete, exercisable, and include evidence of verification and validation. | Functional, significantly exceed minimal functionality. | Results of this paper have been reproduced by a different team using the original artifact. | Results of this paper have been replicated by a different team without the original artifact. |

only 111 out of the 8,774 ICSE research entries were labeled as 'available,' 74 as 'reusable,' 24 as 'functional,' and none as 'replicated' or 'reproduced' reuse (see Table 1 for a definition of those terms). Put another way, according to the ACM Portal, only 2.4% of the ICSE publications are explicitly associated with any kind of reuse. Worse still, according to that report, there were no replicated or reproduced results from ICSE in the last decade.

We argue that the reuse "problem" is more apparent than real—at least in software engineering. We describe a successful approach to recording research reuse where teams of researchers from around the world read 170 recent (2020) conference papers from software engineering. This work generated the "reuse graph" in Figure 1, in which each edge connects papers to the prior work they are (re)using. As we will discuss, when compared to other community monitoring methods (for example, artifact tracks or bibliometric searches[5,12,19]), these reuse graphs require less effort to build and verify. For example, it took around 12 minutes per paper for our team from Hong Kong, Canada, the U.S., Italy, Sweden, Finland, and Australia to apply this reuse graph methodology to software engineering.[a]

The rest of this article discusses generating, applying, and the value of our reuse graphs. Before beginning, we offer the following introductory remark. This article is written as a protest, of sorts, against how we currently assess science and scientific output. This article's authors have worked as researchers for decades, supervising graduate students and organizing prominent conferences and journals. Based on that experience, we assert

a That team included the authors of this paper plus Jacky Keung from City University, Hong Kong; Greg Gay from Chalmers University, Sweden; Burak Turhan from Oulu University, Finland; and Aldeida Aleti from Monash University, Australia. We gratefully acknowledge their work and the work of their graduate students. We especially call out the work of Afonso Fontes from Chalmers University, Sweden.

that researchers do more than write papers. Rather, we are all engaged in long-term stewardship of ideas; as part of that stewardship, we generate more than just papers. Yet, of all our products, it is only our papers that are used, mostly in some annual bibliometric analysis of our worth. We view this as an inadequate way to measure what researchers do.

The problem, we think, is in the very term "bibliometric." This term is heavily skewed toward publications and monographs and the kinds of things we can easily store in the repositories of our professional societies—for example, IEEE Xplore and ACM Portal. In fact, the term "bibliométrie" was first used by Paul Otlet in 1934[25] and was defined as "the measurement of all aspects related to the publication and reading of books and documents."

Subsequent definitions tried to broaden that definition. For example, the anglicized version "bibliometrics" was first used by Alan Pritchard in his 1969 paper, "Statistical Bibliography or Bibliometrics?", where he defined the term as "the application of mathematics and statistical methods to books and other media of communication."[24] But what we are observing in 2022 is that "other media of communication" in software engineering (and other fields) is far broader than just the products stored in the repositories of our professional societies. For example, researchers might use the results of papers, follow guidance from one paper in their own work, or download data or code used on another paper (and then use locally). We argue that all such downloads or guidance-following are examples of reuse, since all are examples of members in our research community reusing products from other research in their own work (for a more exact categorization of the types of reuse we are studying, please see our section Studying Reuse).

It is all too easy to propose a broader definition for how scholars reuse and communicate their products. Such a new definition is practically useless unless we can propose some method to collect data on that new definition. We suggest that our new definitions can be operationalized via crowdsourced methods.

**This article is written as a protest, of sorts, against how we currently assess science and scientific output.**

### Capturing Reuse

There are many methods to map the structure of SE research, such as (a) manual or automatic citation searchers or (b) "artifact evaluation committees" that foster the generation and sharing of research products. Such studies can lag significantly behind current work. For example, in our own prior citation analysis of SE,[19] we only studied up to 2016. The study itself was conducted in 2017, but not fully published till 2018. Given the enormous effort required for that work, we have vowed never to do it again.

Reuse graphs, on the other hand, are faster to update since the work of any individual working on these graphs is minimal. Other reasons for favoring reuse graphs are that they are community comprehensible, verifiable, and correctable. All the data used for our reuse graphs is community-collected and can be audited at https://reuse-dept.org. If errors are detected, issue reports can be raised in our GitHub repository and then corrected. The same may not hold true for studies based on citation servers run by professional bodies and for-profit organizations (see Table 2). New data can be contributed by anyone either directly supplying data in our format or through a user interface directly on our website for easier access. The resulting issue report is then reviewed and, when necessary, corrected. After a third person successfully inspects the data, it is added to the reuse graph.

What is the value of a verified, continually updated snapshot of a current research area? Once our reuse graph covers several years (and not just 2020 conference publications), we foresee several applications:

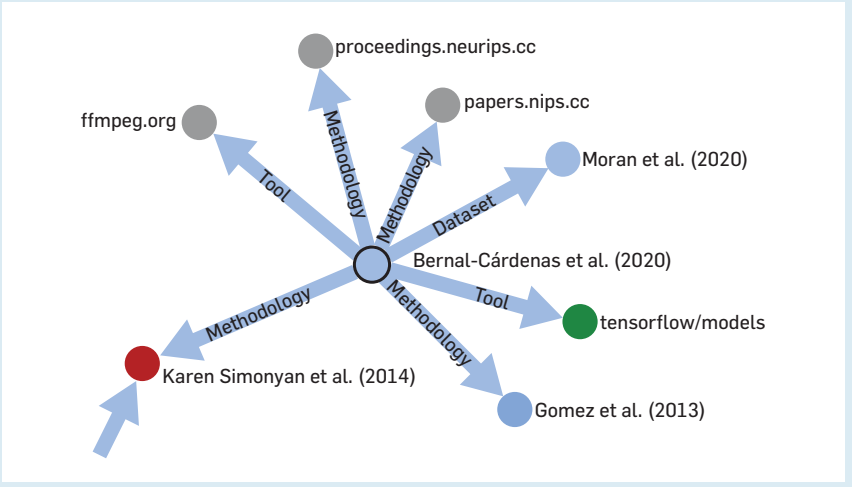
- Academics can check that their contributions to science are being properly recorded.
- When applying for a promotion or new position, research faculty or industrial workers could document the impact of their work beyond papers, including tools, datasets, and innovative methods.
- Graduate students could direct their attention to research areas that are both very new (nodes from recent years) and very productive (nodes with an unusually large number of edges attached).
- Organizers of conferences could

**Table 2. Examples of errors in citation servers.**

| EXAMPLE #1 |
|---|
| One of the authors has an entry in Google Scholar metrics software systems saying that their "How to" paper has received 80 citations over the past five years at IEEE Transactions on Software Engineering (TSE). That link connects to some other paper, not written by any of us. There is no "help" button at Google Scholar where this error can be reported. This is disappointing, since it is suspected that this mysterious "How to" paper references work that might be the most cited from IEEE TSE in the last five years. That would be a significant achievement, if we could document it, but using Google Scholar, we cannot. |
| **EXAMPLE #2** |
| In our recent large-scale text-mining studies of more than 30,000 SE papers,[19] we found that papers can appear on our citation server but not on other servers. Also, examples were found where some papers had 20 times the citations on one server than on another. Again, we were unable to contact anyone working on those citation servers to fix those errors. |
| **EXAMPLE #3** |
| Even if the owners of these citation sites can be contacted, their errors may remain unfixed. For example, there is no accepted convention for how to typeset a hyphen. Different venues used no spaces or a single space around hyphens, while others typeset the hyphen as two dashes. Zhou et al. found that papers with hyphens in the title get reported as different papers in different venues, which means that those papers receive fewer citations.[29] Zhou et al. also reports that when they contacted the owners of these citation servers, rather than fix the errors, the owners lobbied for the Zhou et al. paper not to be published. |

**Figure 2. A detailed view of a section of the reuse graph from Figure 1.**

This figure shows reuse from Bernal-Cárdenas et al.[7] Edges reflect tool, dataset, and methodology reuse. Red nodes indicate arXiv preprint; green represents a GitHub repository; blue denotes a published paper, and grey indicates other websites or grey literature locations. https://www.reuse-dept.org/doi/10.1145/3377811.3380328.




select their keynote speakers from that space of new and productive artifacts.

▸ Growth patterns might guide federal government funding priorities or departmental hiring plans.

▸ Venture capitalists could use these graphs to detect emergent technologies, perhaps even funding some of those.

▸ Conference organizers could check if their program committees have enough members from currently hot topics.

▸ Further, those same organizers could create new conference tracks and journal sections to service active research communities that are underrepresented in current publication venues.

▸ Journal editors could find reviewers with relevant experience.

▸ Educators can use the graphs to guide their teaching plans.

## Studying Reuse

In our reuse study, we targeted papers from the 2020 technical programs of six major international SE conferences: ICSE, Automated Software Engineering (ASE), Joint European Software Engineering Conference/Foundations of Software Engineering (ESEC/FSE), Software Maintenance and Engineering (ICSME), Mining Software Repositories (MSR), and Empirical Software Engineering and Measurement (ESEM). These conferences were selected using advice from Matthew et al.,[19] but our vision is to expand—for example, by looking at all top-ranked SE conferences. GitHub issues were used to divide up the hundreds of papers from those conferences into "work packets" of 10 papers each. Reading teams were set up from software engineering research teams from around the globe in Hong Kong; Istanbul, Turkey; Victoria, Canada; Gothenburg, Sweden); Oulu, Finland; Melbourne, Australia; and Raleigh, NC, USA. Team members assigned themselves work packets and read the papers in search of examples of reuse listed in the next paragraph. Once completed, a second person (from any of our teams) performed the same task and checked for consistency. Fleiss Kappa statistics were then computed to track the level of reader disagreement. GitHub issues[b] were used to manage this in the open, but raters were asked not to examine previous results. A member of this article's author team then performed a final check on disagreements before including the data into the graph.

Teams were asked to record six kinds of reuse:

1. Most papers must benchmark new ideas against some prior recent state-of-the-art paper. That is, they reuse old papers as steppingstones toward new results.

2. Statistical methods are often reused. Here we do not mean "we use a two-tailed t-test" or some other decades-old, widely used statistical method. Rather, we refer to statistical methods in recent papers that propose statistical guidance for the kinds of analysis seen in SE. Perhaps because this kind of analysis is very rare, this work is highly cited. For example:

▸ A 2008 paper, "Benchmarking

b For example, see https://bit.ly/3Vbtqef

Classification Models for Software Defect Prediction,"[18] has 1,178 citations

▸ A 2011 paper, "A Practical Guide for Using Statistical Tests to Assess Randomized Algorithms,"[1] has 778 citations.

3. Metrics and methodology descriptions that are specific to the research area, including software metrics such as CK metrics or flow metrics as well as research methods such as grounded theory or sampling criteria.

4. Datasets

5. Sanity checks, which justify why a particular approach works or is reasonable to avoid bad data—for example, why to avoid using GitHub stars to select repositories.[16]

6. Software packages of the kind currently being reviewed by SE conference AECs (tools and replications).

Figure 2 shows an example. Starting with a paper by Bernal-Cárdenas et al.,[5] we find among others a reused dataset from Moran et al.,[22] tool reuse of FFmpeg and Tensorow object detection, and several reused methods, including the ConvNet approach described by Simonyan. Readers can follow the URL in the Figure 2 caption for more detailed information.

We can report that it is not difficult to read papers to detect these kinds of reuse:

▸ The six types of reuse noted above can be found quickly. Our graduate students report that reading their first paper might take up to an hour. But after two or three papers, the median reading time drops to approximately 12 minutes (see Figure 3a).

▸ When we compare the reuse reported by different readers, we get Figure 3b. In our current results, the median Fleiss Kappa score (for reviewer agreement) is 1—that is, very good.

▸ The one caveat we would add is that graduate students involved in this activity need at least two years of active research experience in their area of study. We base this on the fact that when we tried data collection from a large intro-to-SE graduate subject, the resulting Kappa agreement scores were poor.

The result of this data collection is a directed multi-graph of publications and other forms of dissemination of research artifacts. The edges of this graph are annotated with the type of reuse according to the list above. Reuse metrics for a specific publication (or other form, for example, a GitHub repository) are the in-degree and out-degree measures of the node that represents this publication. When accumulated for the originating authors, individual reuse metrics can be collected. Zooming into the graph on our website, reuse types are visibly annotated at the graph edges (see Figure 2). A filter allows a graph to be extracted for a single reuse type out of the multi-graph.

Of course, there any many more items being reused than just the six we have listed.[c] It is an open question, worthy of future work, to check if those other items can be collected in this way and, indeed, to refine these categories as understanding changes.

## Related Work

Apart from software engineering,[21] many other disciplines are actively engaged in artifact creation, sharing, and reuse.[3,4] Artifacts are useful for building a culture of replication and reproducibility,[9,17] already acknowledged as important in SE.[8,10,15,27] Fields such as psychology have had many early results thrown into doubt due to a failure to replicate the original findings.[28] Sharing research protocols and data through replication packages and artifacts allows for other research teams to conduct severe tests of the original studies,[20] strengthening or rejecting these initial findings.

In medicine, drug companies are mandated to share the research protocols and outcomes of their drug trials, something that has become vitally important recently, albeit not without challenges.[11] In physics and astronomy, artifact sharing is so commonplace that large community infrastructures exist solely to ensure data sharing, not least because the governments which fund these costly experiments insist on it.

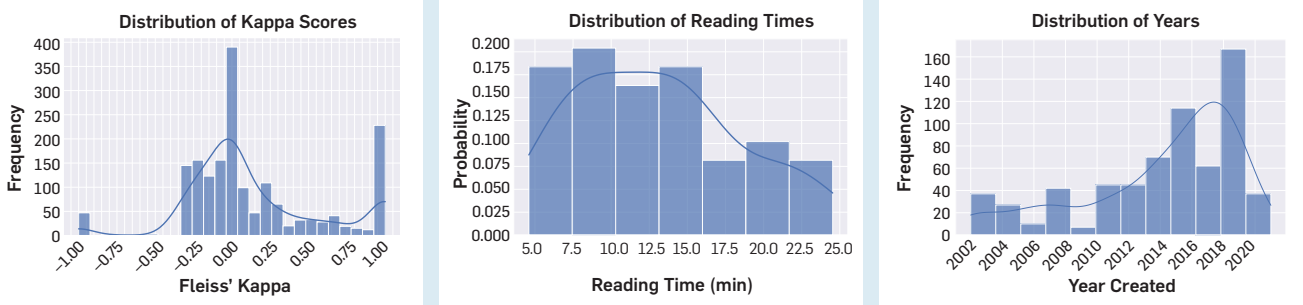
In more theoretical areas of CS, the pioneering use of preprint servers has enabled 'reuse' of proofs, which has been essential to progress. In machine learning, replication is focused on steppingstones, enabled by highly successful benchmarks such as ImageNet.[26] However, recent advances with extremely costly training regimens have called replicability into question.[d]

c For example, see the 22 types of potentially reusable items at https://bit.ly/3FIjMtG

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

**Figure 3. Reading time results, agreement scores, and yearly prevalence of reused papers.**

In the middle chart, the Fleiss Kappa score[13] measures the inter-rater reliability among multiple raters when there are categorical ratings to items. We prefer it to Cohen's kappa measure, which only works for two raters. In our case, we had multiple raters assigning a pool of papers the categories "Yes" (signifying that a paper was reused) or "No" (signifying that it was not). This pool was obtained by taking the union of all the papers marked as being reused by a specific paper.




(a) Reading times (b) Fleiss Kappa scores (c) Years the works were created

In the specific case of SE research, prior to this paper, there was little recorded and verified evidence of reuse. Many researchers have conducted citation studies that find links to highly cited papers—for example, Matthew et al.[19] As stated previously, such studies can lag the latest results. Also, recalling Table 2, we have cause to doubt the conclusions from such citation studies.

From a practical perspective, many conferences have recently introduced AECs to entice reuse and replication. Moreover, authors of accepted conference papers submit software packages that, in theory, let others re-execute that work.[8,9] These committees award badges, as shown in Table 1.

Artifact evaluation is something of a growth industry in the SE as well as the programming languages (PL) communities, as shown in Figure 4, which presents the increasing number of people evaluating artifacts between 2011 and 2019. One may conclude that such practices make the community more aware of what is available and reusable, and therefore, can become a potential node of a reuse graph. As such, the source is explicitly made available to any other researcher willing to (re)use it.[8,14]

Now, the question to be asked is: Are all the people of Figure 4 making the best use of their time? Perhaps not. Most artifacts are assigned the badges requested by the authors, so it might be safe to ask some of the personnel from Figure 4 to, for example, spend less time evaluating conference artifacts and more time working on Figure 1.

But most importantly, it is not clear whether the artifact evaluation process is creating reused artifacts, and therefore, indirectly contributing to the reuse graph concept. Indeed, if we query ACM Portal for "software engineering" and "artifacts" between 2015 and 2020, we find that most of the recorded artifacts are not reused in replications or reproductions.[e] Specifically, only 1/20 are reproduced and only 1/50 are replicated.

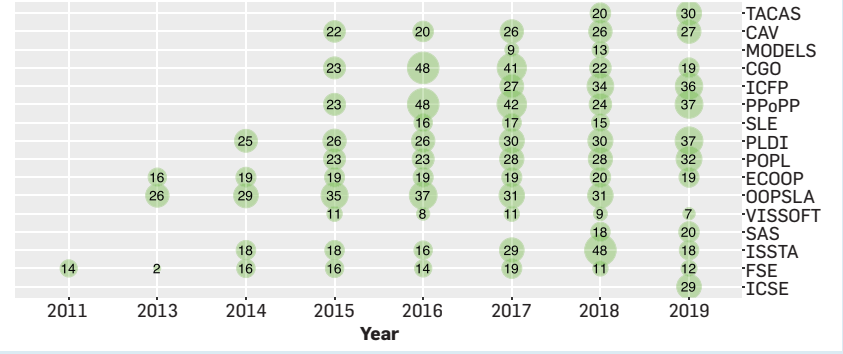
Perhaps it might be useful to reflect more on what is being reused (as we have done earlier in this article). This is what has motivated our research and led us to create the reuse graph.

## The ROSE Initiative

***The Rose Initiative (Recognizing and Rewarding Open Science in Software Engineering) is an international, multi-conference workshop that will continually report updates to the software engineering reuse graphs.***

- **Researchers who reuse the most from other papers will be applauded and awarded an "R-index" (reuse index).**
- **Researchers who build the artifacts that are most reused will be applauded (even louder) and awarded an "R+-index," indicating they are producing the artifacts that are most used by the rest of the community.**
- **Between each conference, the ROSE Initiative will coordinate an international team of volunteers to incrementally update the SE reuse graph. This article's authors have volunteered to serve as the group's initial nucleus, and we will hold open elections to grow that steering committee using others from our community.**
- **This reuse graph will be displayed at a publicly available website (reuse-dept.org), where individual researchers can browse, check their entries, and propose corrections and extensions.**
- **All reuse reports will be double-checked, and disputed claims will then be tripled-checked.**
- **All the tools used to create the site will be freely available for download. Hence, if the SE community does not like how we are running these reuse graphs, they can use the code and data for a different application.**
- **Also, researchers from other disciplines can apply our tools to their own community.**

**Figure 4. Artifact evaluation committee sizes, 2011-2019.[14]**




### Next Steps for Reuse Graphs

When discussing this work with colleagues, we are often asked if we have assessed it. We reply that, at this stage, this is like asking the inventors of kd-trees[6] in 1975 how much that method has sped up commercial databases. Right now, we are engaged in community building and have shown that we can create the infrastructure needed to collect our data with very little effort and not much coding. While Figure 1 is a promising start, scaling up requires that we organize a larger reading population. Our goal is to analyze 200 papers in 2022, 2,000 in 2023, and 5,000 in 2024, by which time we would have covered most of the major SE venues in the last five years. After that, our maintenance goal would be to read around 500 papers per year to keep up to date with the conferences (then, we would move on to journals). Based on Figure 3a, and assuming each paper is read by two people, the maintenance goal would be achievable by a team of 20 people working two hours per month on this task. To organize this work, we have created the ROSE Initiative (see the sidebar: The Rose Initiative for more information).

If that work interests you, then there are many ways you can get involved:

- Visit https://reuse-dept.org if you are a researcher and wish to check that we have accurately recorded your contribution.
- If you want to apply reuse graphs to your community, please use our tools

e As of December 10, 2020, that search returns 2,535 software engineering papers with artifact badges. Of these, 43% are available, 30% are functional, 20% are reusable, 5% are reproduced, and 2% are replicated artifacts.

at https://github.com/bhermann/DoR/.

▸ If you would like to join this initiative and contribute to an up-to-the-minute snapshot of SE research, then please take our how-to-read-for-reuse tutorial,[f] and then visit the dashboard at the GitHub site (bhermann/DoR). Find an issue with no one's face on it, and assign yourself a task.

We see this effort as one part of the broader open science effort, in addition to helping the community identify the state of the art—for example, patterns of growth in the reuse graph. Among the goals of open science are the desire to increase confidence in published results and an acknowledgment that science produces more types of artifacts than just publications: Researchers also produce method innovations, new datasets, and better tools. If we take an agile view of SE science, then as researchers we should focus on generating these artifacts and rapidly securing critique, curation, and clarification from our peers and the public. C

f See https://bit.ly/3hDc2Bf

**If we take an agile view of SE science, then as researchers we should focus on generating artifacts and rapidly securing critique, curation, and clarification.**


**References**

1. Arcuri, A. and Briand, L. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *Proceedings of the 33rd International Conf. on Software Engineering, Association for Computing Machinery* (2011), 1–10; https://doi.org/10.1145/1985793.1985795.
2. Artifact review and badging. Association for Computing Machinery; https://bit.ly/3VaIQz6.
3. Badampudi, D., Wohlin, C., and Gorschek, T. Contextualizing research evidence through knowledge translation in software engineering. *ACM International Conference Proceeding Series* (2019), 306–311; https://doi.org/10.1145/3319008.3319358.
4. Baker, M. and Penny, D. Is there a reproducibility crisis? *Nature 533*, 7604 (2016), 452–454; https://doi.org/10.1038/533452A.
5. Baldassarre, M.T., Caivano, D., Romano, S., and Scanniello, G. Software models for source code maintainability: A systematic literature review. *2019 45th Euromicro Conf. on Software Engineering and Advanced Applications*, IEEE, 252–259.
6. Bentley, J.L. Multidimensional binary search trees used for associative searching. *Communications of the ACM 18*, 9 (September 1975), 509–517; https://doi.org/10.1145/361002.361007.
7. Bernal-Cárdenas, C. et al. Translating video recordings of mobile app usages into replayable scenarios. In *Proceedings of the ACM/IEEE 42nd Intern. Conf. on Software Engineering* (2020); https://doi.org/10.1145/3377811.3380328.
8. Childers, B.R. and Chrysanthis, P.K. Artifact evaluation: Is it a real incentive? *2017 IEEE 13th Intern. Conf. on e-Science*, 488–489; https://bit.ly/3hFa6Z4.
9. Collberg, C. and Proebsting, T.A. Repeatability in computer systems research. *Communications of the ACM 59*, 3 (2016), 62–69; https://bit.ly/3HKABXx.
10. da Silva, F.Q.B. Replication of empirical studies in software engineering research: A systematic mapping study. *Empirical Software Engineering* (September 2012); https://doi.org/10.1007/s10664-012-9227-7.
11. DeVito, N.J., Bacon, S., and Goldacre, B. Compliance with legal requirement to report clinical trial results on ClinicalTrials.gov: A cohort study. *The Lancet 395*, 10221 (2020), 361–369; https://bit.ly/3PDwIFM.
12. Felizardo, K.R. et al. Secondary studies in the academic context: A systematic mapping and survey. *J. of Systems and Software 170* (2020), 110734.
13. Fleiss, J.L. Measuring nominal scale agreement among many raters. *Psychological Bulletin 76*, 5 (1971), 378.
14. Hermann, B., Winter, S., and Siegmund, J. Community expectations for research artifacts and evaluation processes. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering*, Association for Computing Machinery (2020), 469–480; https://doi.org/10.1145/3368089.3409767.
15. Heumüller, R., Nielebock, S., Krüger, J., and Ortmeier, F. Publish or perish, but do not forget your software artifacts. *Empirical Software Engineering* (2020); https://doi.org/10.1007/s10664-020-09851-6.
16. Kalliamvakou, E. et al. The promises and perils of mining GitHub. In *Proceedings of the 11th Working Conf. on Mining Software Repositories—MSR 2014*, ACM Press; https://doi.org/10.1145/2597073.2597074.
17. Krishnamurthi, S. and Vitek, J. The real software crisis: Repeatability as a core value. *Communications of the ACM 58*, 3 (February 2015), 34–36; https://doi.org/10.1145/2658987.
18. Lessmann, S., Baesens, B., Mues, C., and Pietsch, S. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Transactions on Software Engineering 34*, 4 (2008), 485–496; https://bit.ly/3jdmrnF.
19. Mathew, G., Agrawal, A., and Menzies, T. Finding trends in software research. *IEEE Transactions on Software Engineering* (2018), 1–1; https://bit.ly/3hwuUSE.
20. Mayo, D.G. *Statistical Inference as Severe Testing: How to Get Beyond the Statistics Wars.* Cambridge University Press. 2018.
21. Menzies, T. Guest editorial for the Special Section on Best Papers from the 2011 Conference on Predictive Models in Software Engineering (PROMISE). *Information and Software Technology 55*, 8 (2013), 1477–1478.
22. Moran, K. et al. Machine learning-based prototyping of graphical user interfaces for mobile apps. *IEEE Transactions on Software Engineering 46*, 2 (2020), 196–221; https://doi.org/10.1109/tse.2018. 2844788.
23. Popper, K. *Conjectures and Refutations: The Growth of Scientific Knowledge.* Routledge (1963).
24. Pritchard, A. Statistical bibliography or bibliometrics? *J. of Documentation 25*, 4 (1969), 348–349.
25. Rousseau, R. Forgotten founder of bibliometrics. *Nature 510* (2014).
26. Russakovsky, O. et al. ImageNet large scale visual recognition challenge. *Intern. J. of Computer Vision 115*, 3 (2015), 211–252; https://bit.ly/3WkR6hc.
27. Santos, A., Vegas, S., Oivo, M., and Juristo, N. Comparing the results of replications in software engineering. *Empirical Software Engineering 26*, 2 (February 2021); https://doi.org/10.1007/s10664-020-09907-7.
28. Schimmack, U. A meta-psychological perspective on the decade of replication failures in social psychology. *Canadian Psychology 61*, 4 (November 2020), 364–376; https://doi.org/10.1037/cap0000246.
29. Zhou, Z.Q., Tse, T.H., and Witheridge, M. Metamorphic robustness testing: Exposing hidden defects in citation statistics and journal impact factors. *IEEE Transactions on Software Engineering 47*, 6 (2021), 1164–1183; https://bit.ly/3VcqG05.



**Maria Teresa Baldassarre** is a professor at the University of Bari, Italy.

**Neil Ernst** is a professor at the University of Victoria, Canada.

**Ben Hermann** is a professor at the Technische Universität Dortmund, Germany.

**Tim Menzies** (tjmenzie@ncsu.edu) is a professor at North Carolina State University, Raleigh, NC, USA.

**Rahul Yedida** is a Ph.D. candidate at North Carolina State University, Raleigh, NC, USA.




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

DOI:10.1145/3552309

**An examination of how the technology landscape has changed and possible future directions for HPC operations and innovation.**

BY DANIEL REED, DENNIS GANNON, AND JACK DONGARRA

# HPC Forecast: Cloudy and Uncertain

COMPUTING PERVADES ALL aspects of society in ways once imagined by only a few. Within science and engineering, computing has often been called the third paradigm, complementing theory and experiment, with big data and artificial intelligence (AI) often called the fourth paradigm.[14] Spanning both data analysis and disciplinary and multidisciplinary modeling, scientific computing systems have grown ever larger and more complex, and today's exascale scientific computing systems rival global scientific facilities in cost and complexity. However, all is not well in the land of scientific computing.

In the initial decades of digital computing, government investments and the insights from designing and deploying supercomputers often shaped the next generation of mainstream and consumer computing products. Today, that economic and technological influence has increasingly shifted to smartphone and cloud service companies. Moreover, the end of Dennard scaling,[3] slowdowns in Moore's Law, and the rising costs for continuing semiconductor advances have made building ever-faster supercomputers more economically challenging and intellectually difficult.

As Figure 1 suggests, we believe current approaches to designing and constructing leading-edge high-performance computing (HPC) systems must change in deep and fundamental ways, embracing end-to-end co-design; custom hardware configurations and packaging; large-scale prototyping; and collaboration between the dominant computing companies, smartphone and cloud computing vendors, and traditional computing vendors.

We distinguish leading-edge HPC—the very highest-performing systems—from the broader mainstream of midrange HPC. For the latter, market forces continue to shape that market's expansion. Let's begin by examining where the technology landscape has changed and then examine possible future directions for HPC innovation and operations.

## » key insights

- **The future of advanced scientific computing, aka supercomputing or high-performance computing (HPC), is at an important inflection point, being reshaped by a combination of technical challenges and market ecosystem shifts.**
- **These shifts include semiconductor constraints—the end of Dennard scaling, Moore's Law performance limitations, and rising foundry costs—as well ecosystem shifts due to the rise of cloud hyperscalers and the increasing importance of AI technologies.**
- **Building the next generation of leading-edge HPC systems will require rethinking many fundamentals and historical approaches by embracing end-to-end co-design; custom hardware configurations and packaging; large-scale prototyping, as was common 30 years ago; and collaborative partnerships with the dominant computing ecosystem companies.**

IMAGE BY EVANNOVOSTRO

## Ecosystem Shifts

To understand HPC's future potential, one must examine the fundamental shifts in computing technology, which have occurred along two axes: the rise of massive-scale commercial clouds, and the economic and technological challenges associated with the evolution of semiconductor technology.

**Cloud innovations.** Apple, Samsung, Google, Amazon, Microsoft, and other cloud service companies are now major players in the computing hardware and software ecosystems, both in scale and in technical approach. These companies initially purchased standard servers and networking equipment for deployment in traditional collocation centers (colos). As scale increased, they began designing purpose-built data centers optimized for power usage effectiveness (PUE) and deployed at sites selected via multifactor optimization based on the availability of various factors, such as inexpensive energy, tax incentives and political subsidies, political and geological stability, network access, and customer demand.

As cloud scale, complexity, and operational experience continued to grow, additional optimization and opportunities emerged. These include software defined networking (SDN), protocol offloads, and custom network architectures, greatly reducing dependence on traditional network hardware vendors;[6] quantitative analysis of processor,[17] memory,[36,43] network,[7,10] and disk failure modes,[31,35] with consequent redesign for reliability and lower cost (dictating specifications to vendors via consortia such as Open Compute; custom processor SKUs; custom accelerators (FPGAs and ASICs); and complete processor design—for example, Apple silicon, Google TPUs,[19] and AWS Gravitons). In between, the cloud vendors deployed their own global fiber networks.

This virtuous cycle of insatiable consumer demand for rich services, business outsourcing to the cloud, expanding datacenter capacity, and infrastructure cost optimization has had several effects. Most importantly, it has dramatically lessened, and in

**Figure 1. Technical and economic forces reshaping HPC.**

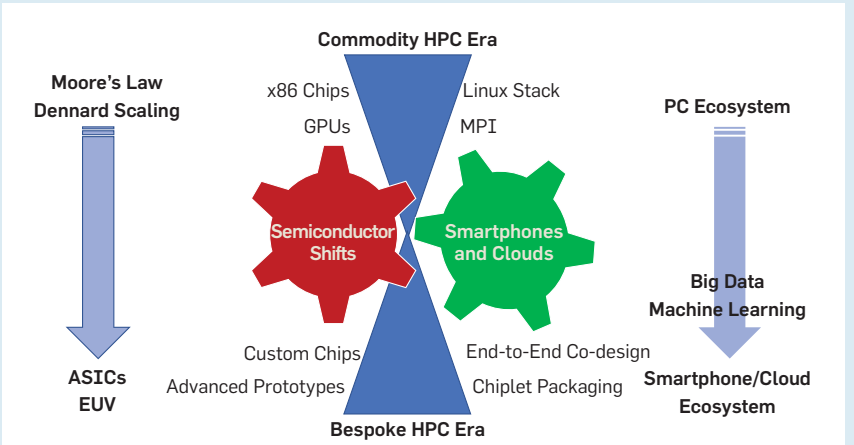
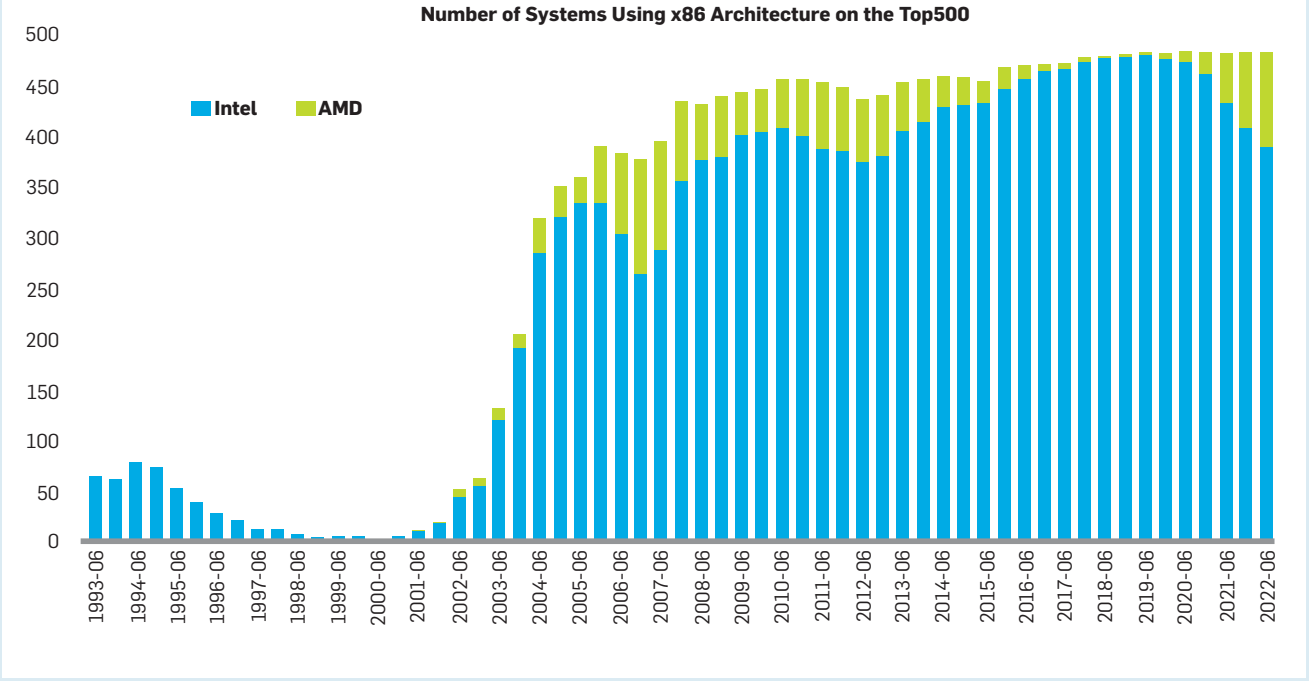



**Figure 2. Systems Using the x86-64 architecture on the TOP500.[39]**



many cases totally eliminated, their dependence on traditional computing vendors. To see the dramatic shifts in influence and scale, one needs to look no further than cloud service-provider and smartphone-vendor market capitalizations, each near or more than 1 trillion USD. Put another way, the locus of innovation and influence has increasingly shifted from chip vendors and system integrators to cloud service providers.

**Semiconductor evolution.** Historically, the most reliable engine of performance gains has been the steady rhythm of semiconductor advances: smaller, faster transistors and larger, higher-performance chips. However, as chip feature sizes have approached 5nm and Dennard scaling has ended,[3] the cadence of new technology generations has slowed, even as semiconductor foundry costs have continued to rise. With the shift to extreme ultraviolet (EUV) lithography[4] and gate-all-around FETs,[5] the "minimax problem" of maximizing chip yields, minimizing manufacturing costs, and maximizing chip performance has grown increasingly complex for all computing domains, including HPC.

Chiplets[a,26,29] have emerged to address these issues, while also integrating multiple functions in a single package. Rather than fabricating a monolithic system-on-a-chip (SoC), chiplet technology combines multiple chips, each representing a portion of the desired functionality, possibly fabricated using different processes by different vendors and including IP from multiple sources. Chiplet designs are part of the most recent offerings from Intel and AMD, where the latter's EPYC and Ryzen processors have delivered industry-leading performance via chiplet integration.[29] Similarly, Amazon's Graviton3 uses a chiplet design with seven different chip dies.

## An HPC Checkpoint

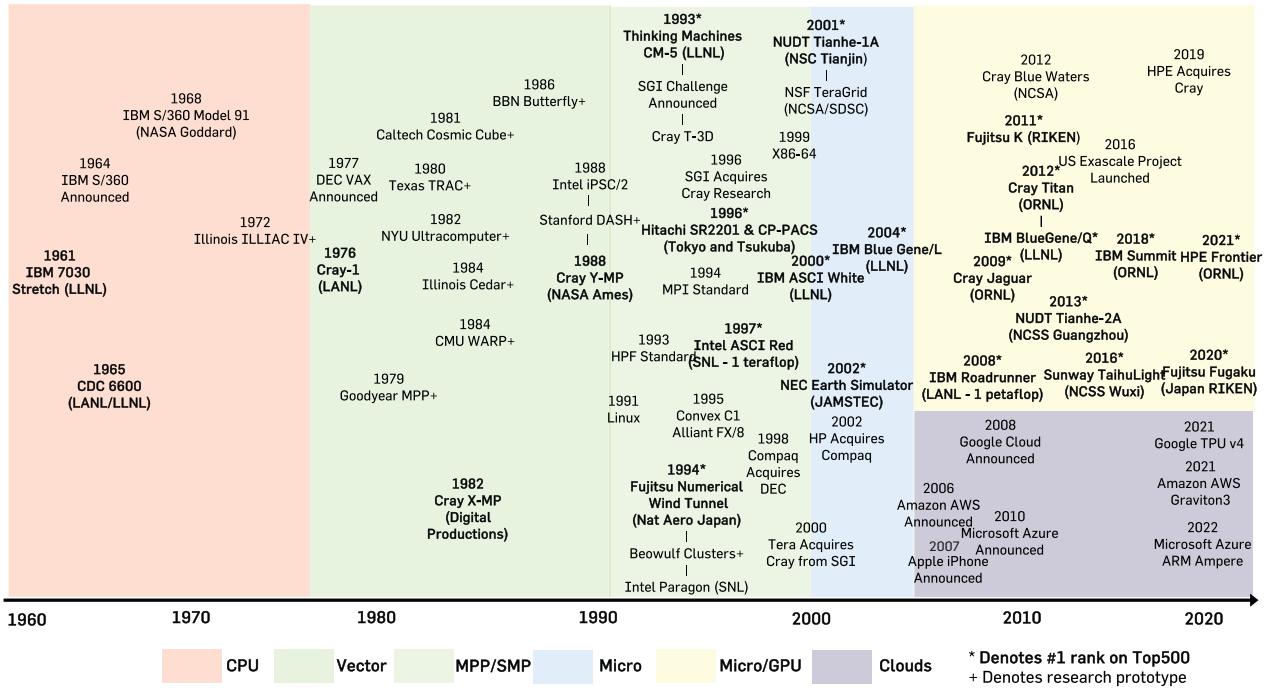
Given the rise of cloud services and increasing constraints on commodity chip performance, it is useful to examine the current state of HPC and how the HPC ecosystem evolved to reach its current structure. From the 1970s to the 1990s, HPC experienced a remarkably active period of architectural creativity and exploration. In the late 1970s, the Cray series of machines[33] introduced vector processing. Companies such as Denelcor and Tera then explored highly multi-threaded parallelism via custom processor design. Universities and companies were also active in exploring new shared memory designs—for example, NYU Ultracomputer,[9] Illinois Cedar,[22] Stanford DASH,[24] and BBN Butterfly.[23]

Finally, distributed-memory, massively parallel computer designs—for example, the Caltech Cosmic Cube,[42] Intel iPSC/2,[40] and Beowulf clusters[38]—established a pattern for hyperscaled performance growth. Riding Moore's Law, the ever-increasing performance of standard microprocessors, together with the cost advantage of volume production, led to the demise of most bespoke HPC systems, a shift often termed the "Attack of the Killer Micros."[27] What followed was academic and industry standardization based on x86-64 processors (see Figure 2) and predominantly gigabit Ethernet and Infiniband networks, the Linux operating system, and message passing via the MPI standard.

By 2000, architectural innovation was limited to node accelerators (for

a See Universal Chiplet Interconnect Express (UCIe) standard; https://www.uciexpress.org

**Figure 3. Timeline of advanced computing.**



example, the addition of GPUs), high-bandwidth memory, and incremental network improvements. The processor, operating system, and network have become the standard interfaces that now define the market boundaries for innovation. At one time, dozens of HPC companies offered competing products. Today, only a few build HPC systems at the largest scales (see Figure 3).

While incremental performance improvements continue with new x86-64 processors and GPU accelerators, basic innovation at the architectural level for supercomputers has largely been lost. However, in the last two years, sparks of architectural creativity are re-emerging, driven by the need to accelerate AI deep learning. Hardware startups, including Graphcore,[18] Groq,[1] and Cerebras,[12] are exploring new architectural avenues. Concurrently, the major cloud service and smartphone providers have also developed custom processor SKUs, custom accelerators (FPGAs and ASICs), and complete processor designs—for example, Apple A15 SoCs, Google TPUs,[19] and AWS Gravitons).

Against this HPC backdrop, the larger computing ecosystem itself is in flux:

- Dennard scaling[3] has ended and continued performance advances increasingly depend on functional specialization via custom ASICs and chiplet-integrated packages.
- Moore's Law is also at or near an end, and transistor costs are likely to increase as feature sizes continue to decrease.
- Advanced computing of all kinds, including HPC, requires ongoing nonrecurring engineering (NRE) investment (that is, endothermic) to develop new technologies and systems.
- The cost to build and deploy leading-edge HPC systems continues to rise, straining traditional acquisition models.
- Smartphone and cloud services companies are cash rich (that is, exothermic); they are designing, building, and deploying their own hardware and software infrastructure at an unprecedented scale.
- AI is fueling a revolution in how both businesses and researchers think about problems and their computational solutions.

**At one time, dozens of HPC companies offered competing products. Today, only a few build HPC systems at the largest scales.**

- Talent is following the money and intellectual opportunities, which are increasingly at a small number of very large companies or creative startups.

With this backdrop, what is the future of computing? Some of it is obvious, given the current dominance of smartphone vendors and cloud service providers. However, it seems likely that continued innovation in advanced HPC will require rethinking some of our traditional approaches and assumptions, including how, where, and when academia, government laboratories, and companies spend finite resources and how we expand the global talent base.

### Leading-Edge HPC Futures

It now seems self-evident that supercomputing, at least at the highest levels, is endothermic, requiring regular infusions of non-revenue capital to fund the NRE costs to develop and deploy new technologies and successive generations of integrated systems. In turn, that capital can come either from other, more profitable divisions of a business or from external sources—for example, government investment. Although most basic computing research is conducted in universities, several large companies (for example, IBM, Microsoft, and Google) still conduct long-term basic research in addition to applied research and development (R&D).

Cloud service companies now offer a variety of HPC clusters of varying size, performance, and price. Given this, one might wonder why cloud service companies are not investing even more deeply in the HPC market. Any business leader must always look at the opportunity cost (that is, the time constant, the talent commitment, and cost of money) for any NRE investments and the expected return on investments. The core business question is always how to make the most money with the money one has, absent some other marketing or cultural reason to spend money on loss leaders, bragging rights, or political positioning. The key phrase here is "the most money;" simply being profitable is not enough, which is why leading-edge HPC is rarely viewed as a primary business opportunity.

The NRE costs for leading-edge

supercomputing are now quite large relative to the revenues and market capitalization of those entities we call "computer companies," and they are increasingly out of reach for most government agencies, at least under current funding envelopes. The days are long past when a few million dollars could buy a Cray-1/X-MP/Y-MP/2 or a commodity cluster, and the resulting system would land in the top 10 of the TOP500 list, a ranking of the world's fastest supercomputers. Today, hundreds of millions of dollars are needed to deploy a machine near the top of the TOP500, and at least similar, if not larger, investments in NRE are needed. In addition, the physical plant and the associated energy and cooling costs for operating such systems are now substantial and continuing to rise. What does this brave new world mean for leading-edge HPC? We believe **five maxims** must guide future HPC government and private sector R&D strategies, for all countries.

**Maxim One: Semiconductor constraints dictate new approaches.** The "free lunch" of lower-cost, higher-performance transistors via Dennard scaling[3] and faster processors via Moore's Law is at an end. Moreover, the de facto assumption that integrating more devices onto a single chip is always the best way to lower costs and maximize performance no longer holds. Individual transistor costs are now flat to rising as feature sizes approach the 1nm range, due to the interplay of chip yields on 300nm wafers and increasing fabrication facility costs. Today, the investment needed to build state-of-the-art facilities is denominated in billions of dollars per facility.

As recent geopolitical events have shown, there are substantial social, political, economic, and national security risks for any country or region lacking a robust silicon fabrication ecosystem. Fabless semiconductor firms rightly focus on design and innovation, but manufacturing those designs depends on reliable access to state-of-the-art fabrication facilities, as the ongoing global semiconductor shortage has shown. The recently passed U.S. CHIPS and Science Act[b] provides roughly 50 billion USD in subsidies to support construction of semiconductor foundries in the U.S., with similar considerations underway in the EU. To date, Intel, Micron, TSMC, and GlobalFoundries have announced plans to build new chip fabrication facilities in the U.S.

Optimization must balance chip fabrication facility costs, now near 10 billion USD at the leading edge; chip yield per wafer; and chip performance. This optimization process has rekindled interest in packaging multiple chips, often fabricated with distinct processes and feature sizes. Such chiplets[26,29] not only enable the mixing of capabilities from multiple sources; they are an economic and engineering reaction to the interplay of chip defect rates, the cadence of feature size reductions, and semiconductor manufacturing costs. However, this approach requires academic, government, and industry collaborations to establish interoperability standards—for example, the Open Domain-Specific Architecture (OSDA) project[41] within the Open Compute Project[c] and the Universal Chiplet Interconnect Express (UCIe)[d] standard. Open chiplet standards can allow the best ideas from multiple sources to be integrated effectively, in innovative ways, to develop next-generation HPC architectures.

**Maxim Two: End-to-end hardware/software co-design is essential.** Leveraging the commodity semiconductor ecosystem has led to an HPC monoculture, dominated by x86-64 processors and GPU accelerators. Given current semiconductor constraints, substantial system performance increases will require more intentional end-to-end co-design,[28] from device physics to applications. China and Japan are developing HPC systems outside the conventional path, as seen by the Top500.

The Fugaku supercomputer[34] (Post-K Computer), developed jointly by RIKEN and Fujitsu Limited based on ARM technology with vector instructions, occupied the top spot on the Top500. It also swept the other rankings of supercomputer performance (HPCG, HPL-AI, and Graph500). Fugaku is designed for versatile use based on a co-design approach between application- and system-development teams.

Likewise, the Chinese government, its academic community, and its domestic HPC vendors have made great efforts in the last few years to build a mature, self-designed hardware and software ecosystem and promote the possibility of running large and complex HPC applications on large, domestically produced supercomputers. It has been reported that China has two exaflops systems (OceanLight and Tianhe-3); several Gordon Bell prize submissions ran on OceanLight.[25] Reflecting global tensions surrounding advanced technologies, a new measure by the U.S. Department of Commerce now precludes companies from supplying advanced computing chips, chip-making equipment, and other products to China unless they receive a special license.

Similar application-driven co-designs were evident in the AI hardware startup companies mentioned previously, as well as the cloud vendor accelerators. Such co-design means more than encouraging tweaks of existing products or product plans. Rather, it means looking holistically at the problem space, then envisioning, designing, testing, and fabricating appropriate solutions. In addition to deep partnerships with hardware vendors and cloud ecosystem operators, end-to-end co-design will require substantially expanded government investment in basic R&D, unconstrained by forced deployment timelines. In addition to partnerships with x86-64 vendors, the ARM license model and the open source RISC-V[11] specification offer intriguing possibilities.

**Maxim Three: Prototyping at scale is required to test new ideas.** Semiconductors, chiplets, AI hardware, cloud innovations—the computing system is now in great flux and not for the first time. As Figure 3 shows, the 1980s and 1990s were filled with innovative computing research projects and companies, many aided by government funding, that built novel hardware, new programming tools,

b See U.S. CHIPS and Science Act; https://science.house.gov/chipsandscienceact

c See Open Compute Project; https://www.opencompute.org/

d See UCIe standard; https://www.uciexpress.org

and system software at large scale. To escape the current HPC monoculture and build systems better suited to current and emerging scientific workloads at the leading edge, we must build real hardware and software prototypes at scale—not just incremental ones, but ones that truly test new ideas using custom silicon and associated software. Implicitly, this means accepting the risk of failure, including at substantial scale, drawing insights from the failure, and building lessons based on those insights. A prototyping project that must succeed is not a research project; it is a product development.

Building such prototypes, whether in industry, national laboratories, or academia, depends on recruiting and sustaining integrated research teams—chip designers, packaging engineers, system software developers, programming environment developers, and application domain experts—in an integrated, end-to-end way. Such opportunities make it intellectually attractive to work on science and engineering problems, particularly given industry partnerships and opportunities to translate research ideas into practice. Implicit in such teams is coordinated funding for workforce development, basic research, and the applied R&D needed to develop and test prototype systems.

**Maxim Four: The space of leading-edge HPC applications is far broader now than in the past.** Leading-edge HPC originated in domains dominated by complex optimization problems and solutions of time-dependent, partial differential equations on complex meshes. Those domains will always matter, but other areas of advanced computing in science and engineering are of high and growing importance. As an example, the *Science 2021 Breakthrough of the Year*[2] was for AI-enabled protein structure prediction,[20] with transformative implications for biology and biomedicine.

Even in traditional HPC domains, the use of AI for dataset reduction and reconstruction, and for PDE solver acceleration, is transforming computational modeling and simulation. Deep-learning methods developed by the cloud companies are changing the course of computational science, and university collaborations are growing. For example, the University of Washington is working with Microsoft Azure on protein-protein interactions.[16] In other areas, OpenAI is showing that deep learning can solve challenging Math Olympiad problems[32] and can also be used to classify galaxies in astrophysics.[21] Generative adversarial networks (GANs)[8] have been used to understand groundwater flow in superfund sites[42] and deep neural networks have been trained to help design non-photonic structures.[30] More than 20 of the papers written for SC21, supercomputing's flagship conference, were on neural networks. The HPC ecosystem is expanding and engaging new domains and approaches in deep learning, creating new and common ground with cloud service providers.

**Maxim Five: Cloud economics have changed the supply-chain ecosystem.** The largest HPC systems are now dwarfed by the scale of commercial cloud infrastructure and social media company deployments. A 500 million USD supercomputer acquisition every five years provides limited financial leverage relative to the billions of dollars spent each year by cloud vendors. Driven by market economics, computing hardware and software vendors, themselves increasingly small relative to the large cloud vendors, now respond most directly to cloud vendor needs.

In turn, government investment (for example, the U.S. Department of Energy's (DOE) Exascale DesignForward, FastForward, and PathForward programs,[e] and the European Union's HPC-Europa3) is small compared to the scale of commercial cloud investments and their leverage with those same vendors. HPC-Europa3, funded under the EU's Eighth Framework Programme, better known as Horizon 2020, has a budget of only 9.2 million euro.[f] Similarly, the U.S. DOE's multiyear investment of 400 million USD via the FastForward, DesignForward, and PathForward programs as part of the Exascale Computing Project (ECP) targeted reduced power consumption, resilience, and improved network and system integration. The DOE only supplied approximately 100 million USD in NRE for each of the exascale systems under construction. During that same period, the cloud companies invested billions. Market research[17] suggests that China, Japan, the U.S., and the EU may each procure one to two exascale-class systems per year, each estimated at approximately 400 million USD.

The financial implications are clear. Government and academic HPC communities have limited leverage and cannot influence vendors in the same ways they did in the past. New, collaborative models of partnership and funding are needed that recognize and embrace ecosystem changes and their implications, both in use of cloud services and collaborative development of new system architectures. The cloud is evolving as a platform where specialized services, such as attached quantum processors, specialized deep-learning accelerators, and high-performance graph database servers, can be configured and integrated into a variety of scientific workflows. However, that is not the whole HPC story. Massive-scale simulations require irregular sparse data structures, and the best algorithms are extremely inefficient on the current generation of supercomputers. The commercial cloud is only a part of HPC's future. New architecture research and advanced prototyping are also needed.

As we have emphasized, the market capitalizations of the smartphone and cloud services vendors now dominate the computing ecosystem, and the overlap between commercial AI application hardware needs and those of scientific and engineering computing is creating new opportunities. We realize this may be heretical to some, but there are times and places where commercial cloud services can be the best option to support scientific and engineering computing needs.

The performance gaps between cloud services and HPC gaps have lessened substantially over the past

e See DoE/Exascale Computing Project's Pathforward; https://www.exascaleproject.org/research-group/pathforward/

f See the European Commission's "Transnational Access Programme for a Pan-European Network of HPC Research Infrastructures and Laboratories for Scientific Computing"; https://cordis.europa.eu/project/id/730897

decade, as shown by a recent comparative analysis.[13] Moreover, HPC as a service is now real and effective, both because of its performance and the rich and rapidly expanding set of hardware capabilities and software services. The latter is especially important; cloud services offer some features not readily available in the HPC software ecosystem.

Some in academia and the national laboratory community will immediately say, "But, we can do it cheaper, and our systems are bigger!" Perhaps, if one looks solely at retail prices, but those may not be the appropriate perspectives. Proving such claims means being dispassionate about technological innovation, NRE investments, and opportunity costs. In turn, this requires a mix of economic and cultural realism in making build versus use decisions and taking an expansive view of the application space, unique hardware capabilities, and software tools. Opportunity costs are real, though not often quantified in academia or government. Today, capacity computing (that is, solving an ensemble of smaller problems) can easily be satisfied with a cloud-based solution, and on-demand, episodic computing of both capacity and large-scale scientific computing can benefit from cloud scaling.

## Conclusion

The computing ecosystem is in enormous flux, creating both opportunities and challenges for the future of advanced scientific computing. For the past 20 years, the most reliable engine of HPC performance gains has been the steady improvement in commodity CPU technology due to semiconductor advances. But with the slowing of Moore's Law and the end of Dennard scaling, improved performance of supercomputers has increasingly relied on larger scale (that is, building systems with more computing elements) and GPU co-processing. Concurrently, the computing ecosystem has shifted, with the rise of hyperscale cloud vendors that are developing new hardware and software technologies.

Looking forward, it seems increasingly unlikely that future high-end HPC systems will be procured and assembled solely by commercial integrators from only commodity components. Rather, future advances will require embracing end-to-end design, testing, evaluating advanced prototypes, and partnering strategically with not only traditional chip and HPC vendors but also with the new cloud ecosystem vendors. These are likely to involve collaborative partnerships among academia, government laboratories, chip vendors, and cloud providers; increasingly bespoke systems designed and built collaboratively to support key scientific and engineering workload needs; or a combination of these two.

**Simply being profitable is not enough, which is why leading-edge HPC is rarely viewed as a primary business opportunity.**

Put another way, in contrast to midrange systems, leading-edge HPC systems are increasingly similar to large-scale scientific instruments (for example, the Vera Rubin Observatory, the LIGO gravity wave detector, or the Large Hadron Collider), with limited economic incentives for commercial development. Each contains commercially designed and constructed technology, but each also contains large numbers of custom elements for which there is no sustainable business model. Instead, we build these instruments because we want them to explore open scientific questions, and we recognize that their design and construction requires both government investment and innovative private sector partnerships.

Like many other large-scale scientific instruments, where international collaborations are an increasingly common way to share costs and facilitate research collaborations, leading-edge computing would benefit from increased international partnerships, recognizing that in today's world, national security and economic competitiveness issues will necessarily limit sharing certain "dual-use" technologies. Subject to those very real constraints, if we are to build more performant, leading-edge HPC systems, we believe there is a need for greater government investment in semiconductor futures—both basic research and foundry construction—along with an integrated, long-term R&D program that funds academic, national laboratory, and private-sector partnerships to design, develop, and test advanced computing prototypes.

These investments must be tens, perhaps hundreds of billions of dollars, in scale.

We have long relied on the commercial market for the building blocks of leading-edge HPC systems. Although this has leveraged commodity economics, it has also resulted in systems ill-matched to the algorithmic needs of scientific and engineering applications. With the end of Moore's Law, we now have both the opportunity and the pressing need to invest in first principles design.

Investing in the future is never easy, but it is critical if we are to continue to develop and deploy new generations of HPC systems, ones that leverage economic shifts, commercial practices, and emerging technologies to advance scientific discovery. Intel's Andrew Grove was right when he said, "Only the paranoid survive," but paranoia alone is not enough. Successful competitors also need substantial financial resources and a commitment to technological opportunities and scientific innovation.

## Acknowledgments

Jack Dongarra was supported in part by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. The authors are grateful for thoughtful feedback on earlier drafts of this article from the anonymous reviewers, as well as Doug Burger (Microsoft), Tony Hey (U.K. Science and Technology Facilities Council), and John Shalf (Lawrence Berkeley National Laboratory). C

**References**


1. Abts, D. et al. Think fast: A Tensor streaming processor (TSP) for accelerating deep learning workloads. In *Proceedings of the ACM/IEEE 47th Annual Intern. Symp. on Computer Architecture*, IEEE Press (2020), 145–158; http://bit.ly/3PA2a87.
2. Beckwith, W. Science's 2021 breakthrough: AI-powered protein prediction. *AAAS* (December 2021); http://bit.ly/3W4j6Gd.
3. Bohr, M. A 30 year retrospective on Dennard's MOSFET scaling paper. *IEEE Solid-State Circuits Society Newsletter 12*, 1 (2007), 11–13; https://doi.org/10.1109/N-SSC.2007.4785534.
4. Chang, Y-W., Liu, R-G., and Fang, S-Y. EUV and e-beam manufacturability: Challenges and solutions. In *Proceedings of the 52nd Annual Design Automation Conf*, Association for Computing Machinery (June 2015), 1-6; https://doi.org/10.1145/2744769.2747925.
5. Dey, S. et al. Performance and opportunities of gate-all-around vertically stacked nanowire transistors at 3nm technology nodes. In *2019 Devices for Integrated Circuit*, 94–98; https://doi.org/10.1109/DEVIC.2019.8783385.
6. Firestone, D. et al. Azure accelerated networking: SmartNICs in the public cloud. *15th USENIX Symp. on Networked Systems Design and Implementation 15*. (2015); https://bit.ly/3HFyNzd.
7. Gill, P., Jain, N., and Nagappan, N. Understanding network failures in data centers: Measurement, analysis, and implications. In *Proceedings of the ACM SIGCOMM 2011 Conf*, 350–361; https://doi.org/10.1145/2018436.2018477.
8. Goodfellow, I. et al. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27* (2014), Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (Eds.), Curran Associates, Inc.; https://bit.ly/3BI6IDy.
9. Gottlieb, A. et al. The NYU Ultracomputer—Designing a MIMD, shared-memory parallel machine. In *Proceedings of the 9th Annual Symp. on Computer Architecture, IEEE Computer Society Press* (1982), 27–42.
10. Govindan, R. et al. Evolve or die: High-availability design principles drawn from Google's network infrastructure. In *Proceedings of the 2016 ACM SIGCOMM Conf*., 58–72; https://doi.org/10.1145/2934872.2934891.
11. Greengard, S. Will RISC-V revolutionize computing? *Communications of the ACM 63*, 5 (April 2020), 30–32; https://doi.org/10.1145/3386377.
12. Groeneveld, P. Wafer scale interconnect and pathfinding for machine learning hardware. In *Proceedings of the Workshop on System-Level Interconnect: Problems and Pathfinding Workshop* (2020); https://doi.org/10.1145/ 3414622.3432992.
13. Guidi, G. et al. 10 years later: Cloud computing is closing the performance gap. *Companion of the ACM/SPEC Intern. Conf. on Performance Engineering* (2021), 41–48; https://bit.ly/3FBFa3I.
14. Hey, T., Tansley, S., and Tolle, K. (Eds.). *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research (2009); https://bit.ly/3v3qXrn.
15. Hochschild, P.H. et al. Cores that don't count. In *Proceedings of the Workshop on Hot Topics in Operating Systems*. Association for Computing Machinery (2021), 9–16; https://bit.ly/3hxVSZT.
16. Horvitz, E. A leap forward in bioscience (2022); https://erichorvitz.com/Leap_forward_bioscience.htm.
17. HPC market update briefing during SC21. *Supercomputing 2021*; https://bit.ly/3HR6Kg2.
18. Jia, Z., Tillman, B., Maggioni, M., and Scarpazza, D.P. Dissecting the Graphcore IPU architecture via microbenchmarking. *CoRR*, abs/1912.03413 (2019). arXiv:1912.03413; http://arxiv.org/abs/1912.03413.
19. Jouppi, N.P. et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual Intern. Symp. on Computer Architecture*, Association for Computing Machinery (2017), 1–12; https://doi.org/10.1145/3079856.3080246.
20. Jumper, J. et al. Highly accurate protein structure prediction with AlphaFold. *Nature* (August 2021), 1476–4687; https://doi.org/10.1038/s41586-021-03819-2.
21. Kim, E.J. and Brunner, R.J. Star-galaxy classification using deep convolutional neural networks. *Monthly Notices of the Royal Astronomical Society 464*, 4 (2016).
22. Kuck, D.J., Davidson, E.S., Lawrie, D.H., and Sameh, A.H. Parallel supercomputing today and the cedar approach. *Science 231*, 4741 (1986), 967–974; https://doi.org/10.1126/science.231.4741.967; arXiv:https://www.science.org/doi/pdf/10.1126/science.231.4741.967.
23. LeBlanc, T.J., Scott, M.L., and Brown, C.M. Large-scale parallel programming: Experience with BBN Butterfly parallel processor. *SIGPLAN Not. 23*, 9 (January 1988), 161–172; https://doi.org/10.1145/62116.62131.
24. Lenoski, D. et al. The Stanford DASH Multiprocessor. *Computer 25*, 3 (March 1992), 63–79; https://doi.org/10.1109/2.121510.
25. Liu, Y. et al. Closing the "quantum supremacy" gap: Achieving real-time simulation of a random quantum circuit using a new Sunway supercomputer. In *Proceedings of the Intern. Conf. for High Performance Computing, Networking, Storage and Analysis*, Association for Computing Machinery (2021); https://doi.org/10.1145/3458817.3487399.
26. Loh, G.H., Naffziger, S., and Lepak, K. Understanding chiplets today to anticipate future integration opportunities and limits. In *2021 Design, Automation Test in Europe Conf. Exhibition*, 142–145; https://doi.org/10.23919/DATE51398.2021.9474021.
27. Markoff, J. The attack of the killer micros. *The New York Times* (May 6, 1991); https://bit.ly/3FDzhDr.
28. Murray, C. et al. Basic research needs for microelectronics. U.S. DoE Office of Science, (October 2018); https://doi.org/10.2172/1616249.
29. Naffziger, S. et al. Pioneering chiplet technology and design for the AMD EPYC™ and Ryzen™ processor families: Industrial product. *2021 ACM/IEEE 48th Annual Intern. Symp. on Computer Architecture*, 57–70; https://doi.org/10.1109/ISCA52012.2021.00014.
30. Peurifoy, J. et al. Nanophotonic particle simulation and inverse design using artificial neural networks. *Science Advances 8*, 5 (2022).
31. Pinheiro, E., Weber, W-D., and Barroso, L.A. Failure trends in a large disk drive population. In *Proceedings of the 5th USENIX Conf. on File and Storage Technologies*, USENIX Association, (2007), 2.
32. Polu, S., Han, J.M., and Sutskever, I. Solving (some) formal math Olympiad problems. Open AI (2022); https://openai.com/blog/formal-math.
33. Russell, R.M. The CRAY-1 computer system. *Communications of the ACM 21*, 1 (January 1978), 63–72; https://doi.org/10.1145/359327.359336.
34. Sato, M. et al. Co-design for A64FX Manycore processor and "Fugaku". In *Proceedings of the Intern. Conf. for High Performance Computing, Networking, Storage and Analysis*, IEEE Press (2020).
35. Schroeder, B. and Gibson, G.A. Understanding disk failure rates: What does an MTTF of 1,000,000 hours mean to you? *ACM Trans. Storage 3*, 3 (October 2007), 8–es; https://doi.org/10.1145/1288783.1288785.
36. Schroeder, B., Pinheiro, E., and Weber, W-D. DRAM errors in the wild: A large-scale field study. *Communications of the ACM 54*, 2 (February 2011), 100–107; https://doi.org/10.1145/1897816.1897844.
37. Seitz, C.L. The cosmic cube. *Communications of the ACM 28*, 1 (1985), 22–33; https://bit.ly/3FWoLIy.
38. Sterling, T. et al. BEOWULF: A parallel workstation for scientific computation. In *Proceedings of the 24th Intern. Conf. on Parallel Processing 1*, CRC Press (1995), I:11–14.
39. Strohmaier, E., Meuer, H.W., Dongarra, J., and Simon, H.D. The TOP500 list and progress in high-performance computing. *Computer 48*, 11 (2015), 42–49; https://doi.org/10.1109/MC.2015.338.
40. The Intel iPSC/2 system: The concurrent supercomputer for production applications. In *Proceedings of the 3rd Conf. on Hypercube Concurrent Computers and Applications: Architecture, Software, Computer Systems, and General Issues 1*, Association for Computing Machinery (January 1988), 843–846; https://doi.org/10.1145/62297.62412.
41. Vinnakota, B. The open domain-specific architecture: Next steps to production. In *Proceedings of the 8th Annual ACM Intern. Conf. on Nanoscale Computing and Communication*. Association for Computing Machinery (2021); https://bit.ly/3PBPtK0.
42. Yang, L. et al. Highly scalable, physics-informed GANs for learning solutions of stochastic PDEs. *ArXiv* abs/1910.13444v1 (2021).
43. Zivanovic, D. et al. DRAM errors in the field: A statistical approach. In *Proceedings of the Intern. Symp. on Memory Systems*, Association for Computing Machinery (2019), 69–84; https://bit.ly/3HI3EuR.


**Daniel Reed** is a Presidential Professor at the University of Utah, Computer Science and Electrical & Computer Engineering, Salt Lake City, Utah, USA.

**Dennis Gannon** is Professor Emeritus at the Indiana University, Luddy School of Informatics, Computing and Engineering, Bloomington, Indiana, USA.

**Jack Dongarra** is Professor Emeritus at the University of Tennessee, Innovative Computing Laboratory, EECS Department, Knoxville, Tennessee, USA.

**ACM BOOKS**

**Collection II**

This book discusses two questions in Complexity Theory: the Monotonicity Testing problem and the 2-to-2 Games Conjecture.

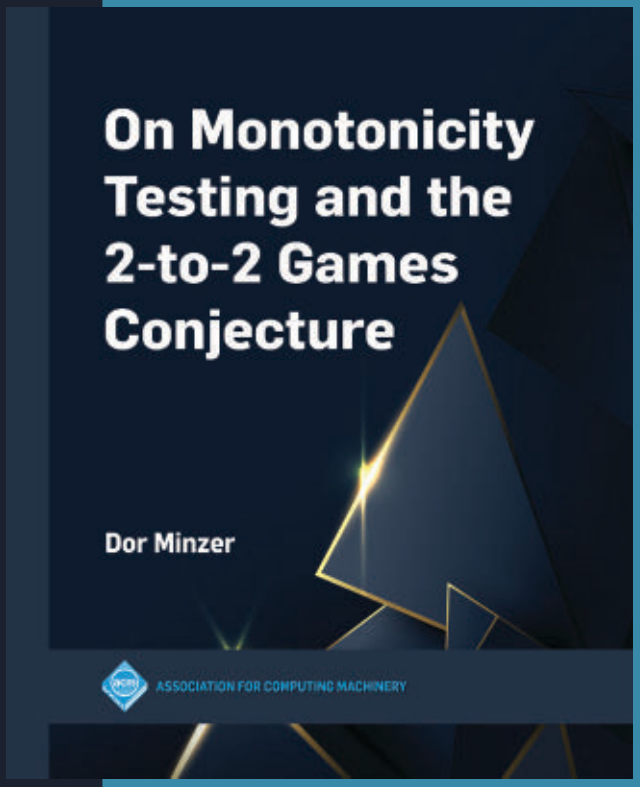
Monotonicity testing is a problem from the field of property testing, first considered by Goldreich et al. in 2000. The input of the algorithm is a function, and the goal is to design a tester that makes as few queries to the function as possible, accepts monotone functions and rejects far-from monotone functions with a probability close to 1.

The first result of this book is an essentially optimal algorithm for this problem. The analysis of the algorithm heavily relies on a novel, directed, and robust analogue of a Boolean isoperimetric inequality of Talagrand from 1993.

The probabilistically checkable proofs (PCP) theorem is one of the cornerstones of modern theoretical computer science. One area in which PCPs are essential is the area of hardness of approximation. Therein, the goal is to prove that some optimization problems are hard to solve, even approximately. Many hardness of approximation results were proved using the PCP theorem; however, for some problems optimal results were not obtained. This book touches on some of these problems, and in particular the 2-to-2 games problem and the vertex cover problem.

The second result of this book is a proof of the 2-to-2 games conjecture (with imperfect completeness), which implies new hardness of approximation results for problems such as vertex cover and independent set. It also serves as strong evidence towards the unique games conjecture, a notorious related open problem in theoretical computer science. At the core of the proof is a characterization of small sets of vertices in Grassmann graphs whose edge expansion is bounded away from 1.

**http://books.acm.org**
**http://store.morganclaypool.com/acm**




# On Monotonicity Testing and the 2-to-2 Games Conjecture

**Dor Minzer**, *Massachusetts Institute of Technology*

ISBN: 9781450399661
DOI: 10.1145/3568031

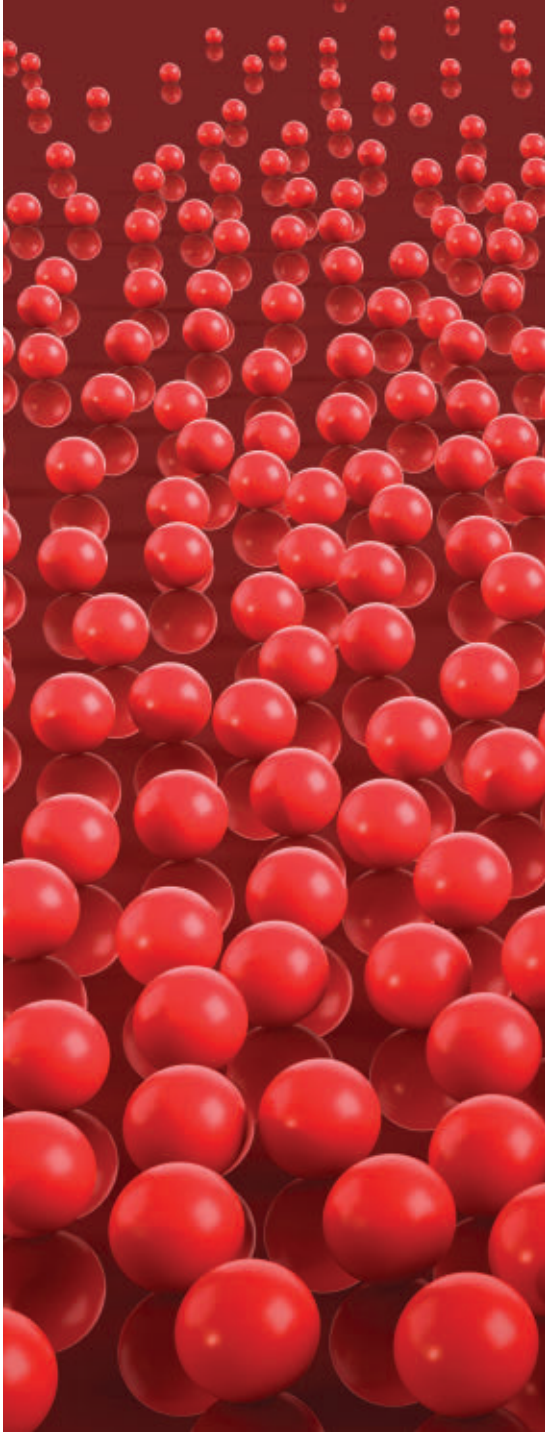
DOI:10.1145/3551635

**A taxonomy of the methods used to obtain quality datasets enhances existing resources.**

BY CHEN SHANI, JONATHAN ZARECKI, AND DAFNA SHAHAF

# The Lean Data Scientist:

## Recent Advances toward Overcoming the Data Bottleneck



OBTAINING DATA HAS become the key bottleneck in many machine-learning (ML) applications. The rise of deep learning has further exacerbated this issue. Although high-quality ML models are finally making the transition from expensive-to-develop, highly specialized code to something more like a commodity, these models involve millions (or even billions) of parameters and require massive amounts of data to train. Thus, the dominant paradigm in ML today is to create a new (large) dataset whenever facing a novel task. In fact, there are now entire conferences dedicated to the creation of new data resources (for example, the International Conference on Language Resources and Evaluation or resource papers at CIKM).

While this approach resulted in significant advances, it suffers from a major caveat, as collecting large, high-quality datasets is often very demanding in terms of time and human resources. For several tasks, such as rare disease detection, large datasets are nearly infeasible to construct.

While there has been much effort suggesting workarounds to this *data-bottleneck* problem, they are scattered across many different subfields, often unaware of one another. There exist many method-specific and domain-specific surveys, but broader, big-picture surveys are difficult to find. The closest in spirit to our work is Roh et al.,[33] which focuses more

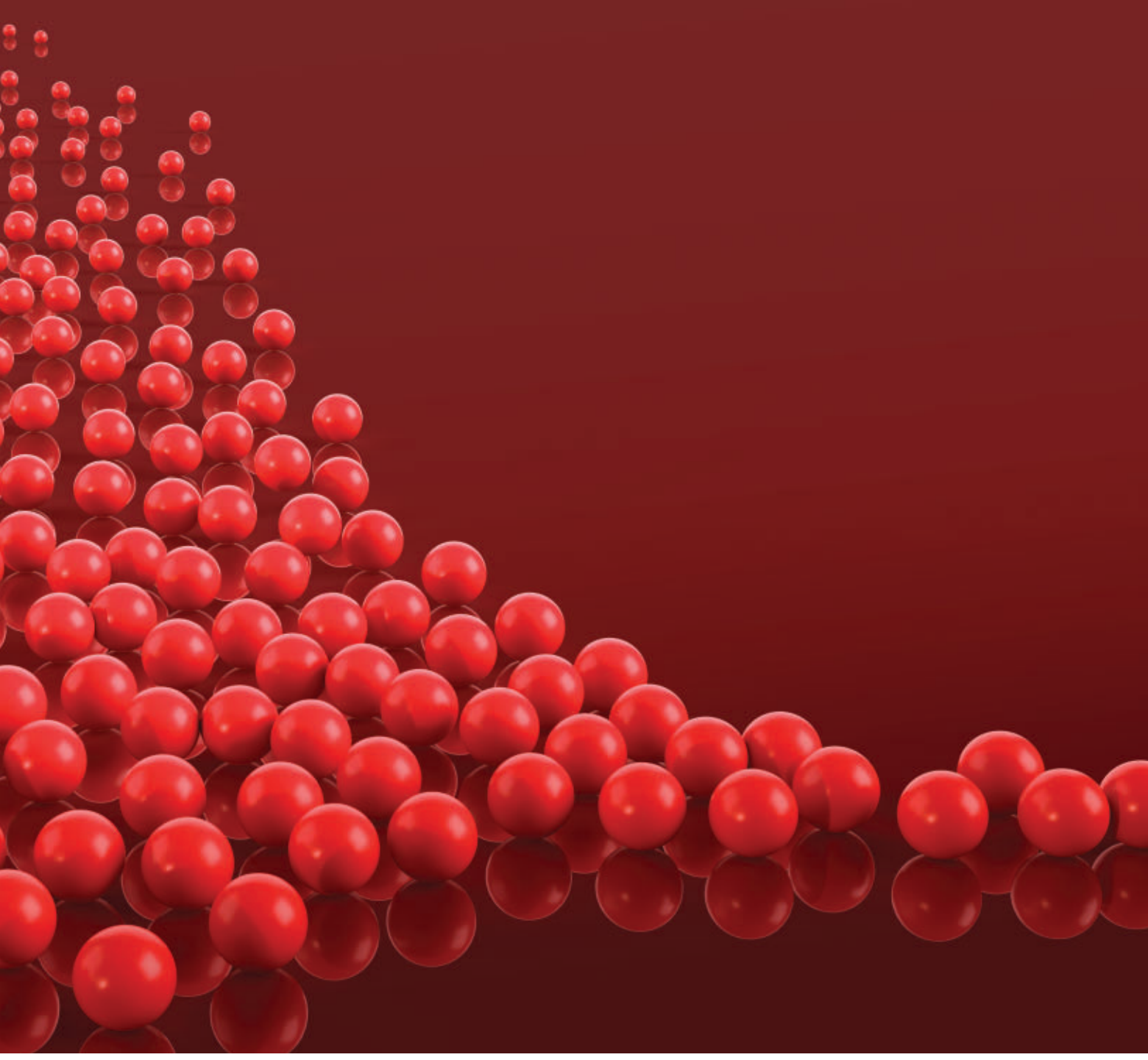


IMAGE BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES

on the data management point of view and the early stages of the pipeline.

In this article, we aim to bring order to this area. Our main contribution is a simple yet comprehensive taxonomy of ways to tackle the data bottleneck. We survey major research directions and organize them into a taxonomy in a way designed to be useful for practitioners choosing between different approaches. The emphasis here is not on covering methods in depth; rather, we discuss the main ideas behind various methods, the assumptions they make and their underlying concepts. For each topic, we mention several important or interesting works, and refer the interested reader to surveys where possible.

We wish to first *raise awareness* of the methods that already exist, to encourage more efficient use of data. In addition (and perhaps more importantly), we hope the organization of the taxonomy would also reveal gaps in current techniques and suggest novel directions of research that could inspire the creation of new, less data-hungry learning methods.

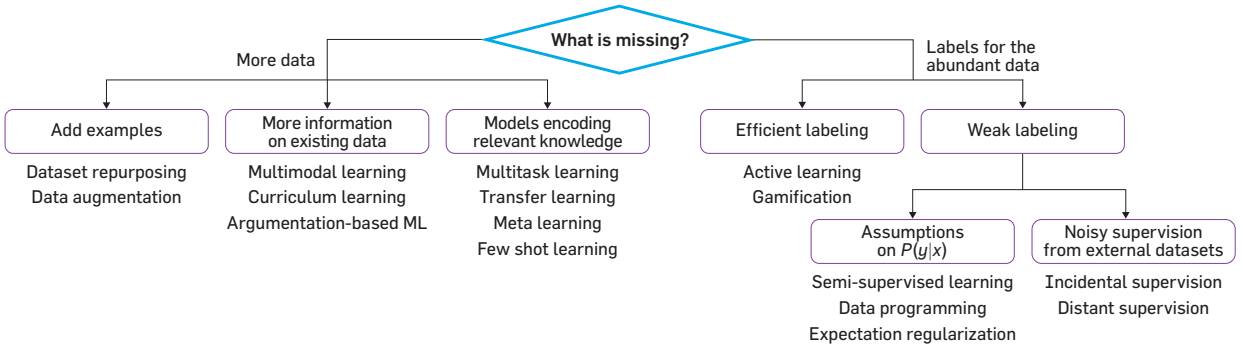
## Taxonomy

**A note on scope.** The data-bottleneck problem is widespread across the field of machine learning. It is especially crucial in supervised learning but applies

### » key insights

- **Recent machine learning algorithms are increasingly data-hungry. A widespread approach is to construct large, task-specific datasets, which is inefficient and sometimes infeasible.**
- **Many ways to tackle this data bottleneck problem have been proposed, but they are scattered across different subfields.**
- **We present a practitioner-centric taxomony of these methods. We distill each method's main assumptions and explain when it is useful, in hopes of encouraging more efficient use of resources as well as uncovering novel research directions.**

**Figure 1. Flowchart of the taxonomy for ways to tackle the data bottleneck.**




to unsupervised paradigms as well. In this work we focus on the supervised, unsupervised, and semi-supervised settings. Reinforcement learning is generally beyond the scope of this article, although some of the methods we present are applicable to it.

We start with a high-level view of our taxonomy, depicted in Figure 1. We first make the distinction between cases where *data* ($X$) is hard to collect, and cases where *labels* ($Y$) pose the difficulty. For example, collecting a dataset of patients with rare diseases is challenging due to the condition's rarity. In contrast, it is relatively easy to collect a large dataset of unlabeled images for an image segmentation task, but annotation is slow and costly.

If obtaining data is the main obstacle, we identify three major approaches:

- **Add examples:** Generate more examples from available data (for example, through data augmentation).
- **Use additional information on existing data:** Increase the dimensionality of $X$ in a manner that can assist the learner (for example, curriculum learning).
- **Use models encoding relevant knowledge:** Instead of learning from scratch, take advantage of models trained in a different yet relevant setup (for example, transfer learning).

If unlabeled data is abundant but labels are difficult to obtain, we identify two main approaches:

- **Acquire labels efficiently:** Label examples that should heavily contribute to the learning process.
- **Weak labeling:** Using proxy labels, either making assumptions about label distribution (for example, semi-supervised learning) or about the labeling process (for example, data programming), or using external (noisy) supervision signals correlated with the true labels (incidental supervision).

We note these approaches may also be combined. For example, one might add more examples and increase the dimensionality of the data. Here, we follow the taxonomy and elaborate on the different approaches and best practices.

## Obstacle: Missing Data

Quite often, data is hard (or impossible) to obtain. In the following, we survey some of the main methods from the left branch in Figure 1: obtaining more examples efficiently, adding informative dimensions to existing data, or taking advantage of related tasks.

Add examples. This category focuses on methods for obtaining more examples.

### Dataset repurposing

**Use a preexisting dataset for a new purpose.**

*Dataset repurposing* is perhaps the most obvious method to add data and is mentioned here for the sake of completeness. The idea is to use a preexisting dataset for a different task than it was originally constructed for.

For example, ImageNet was originally made and used for classification, but later was reused for image generation.[45] Similarly, the MS-COCO image captioning dataset was reused for training visually grounded word embeddings.[20]

Data repurposing also includes *transformations* on existing datasets. For example, consider *inpainting*, the process of restoring lost parts of an image based on the surrounding information. Inpainting is done using various preexisting datasets such as CelebA, Place2, and ImageNet,[39] where the same image splits into both $X$ and $Y$ (sometimes in more than one way).

Of course, it is also possible to repurpose a dataset created with no machine-learning task in mind at all: for example, Bertero and Fung[6] used a dataset of TV sitcoms for a supervised humor detection task, with recorded laughter serving as labels.

### Data augmentation

**Perform transformations on $X$ to enlarge the dataset.**

*Data augmentation* is a common approach for generating more data; it artificially inflates the training set by applying modifications. This method's initial goal was to prevent overfitting.

Data augmentation often employs *vicinal risk minimization* (VRM).[48] In VRM, human knowledge is needed to define a neighborhood around each example in the training data, and virtual examples are drawn from this vicinity distribution. It is easiest to demonstrate this idea in the field of computer vision; there, common augmentations are geometric transformations such as flipping, cropping, scaling, and rotating (see Figure 2). The idea is to make the classifier invariant to change in position and orientation. Similarly, photometric transformations amend the color channels to make the classifier invariant to change in lighting and color.

Data augmentation leads to improved generalization, especially with small datasets[3] or when the dataset is unbalanced (instead of under sampling, which is data-inefficient).

Augmentation methods have seen

a recent surge of interest. Recent advances include methods that jointly train a model for generating augmentations,[28] and methods that learn which augmentations best fit the data.[7] For example, AutoAugment[7] randomly chooses a sub-policy of batch transformation and searches for the one that yields the highest validation accuracy.

Beyond human-defined transformations, recent methods suggested using pretrained generative adversarial networks (GANs) to create new examples. Interestingly, the generated data points do not have to be *interpretable* by humans. For example, Mixup[59] trains a neural network on convex combinations of pairs of examples and their interpolated labels, treating it as "noisy" training data.

**More information on existing data.** Instead of adding new data points, this set of methods focuses on adding dimensions to existing points.

## Multimodal learning

**Integrate associated information on X from multiple modalities.**

*Multimodal learning* attempts to enrich the input to the learning algorithm, giving the learner access to more than one modality of $X$; for example, an image accompanied by its caption. Multimodal learning is intuitive and like how infants learn (that is, children see new objects is often accompanied by additional semantic information). The main drawbacks of multimodal learning are obtaining rich input and effectively integrating it into the model.

Although the term "multimodal learning" is recent, many works combined information from different modalities.[11,22,41] These works, and more recent ones, show the promise of this method as an effective way to reduce data requirements and improve generalization.

Moreover, multimodal learning is also often used when the number of data points is extremely small, and in particular, few-, one-, and zero-shot learning (when only a few target-specific labeled examples exist for the learning process; thus, the learner must understand new concepts using only a handful of examples). For example, Visotsky et al.[51] used multimodal learning for few-shot learning by integrating additional per-sample information—in this case, a list of objects appearing in the input image (see Figure 3). Schwartz et al.[37] demonstrated that it is possible to outperform previous state of the art results on the popular miniImageNet and CUB few-shot learning benchmarks by combining images with multiple and richer semantics (category labels, attributes, and natural language descriptions).

## Curriculum learning

**Present examples to the learner according to a predetermined order, usually based on difficulty.**

In *curriculum learning*, the learner is exposed to examples using a predetermined curriculum, where examples are usually sorted in increasing order of difficulty. Meta-data on $X$ is needed to determine its place in the learning process.

The motivation behind curriculum learning comes from humans, as teachers tend to start by teaching simpler concepts (for example, learning to ride a bicycle with training wheels first). Thus, curriculum learning attempts to augment training examples with a difficulty score, often corresponding to *typicality*.

Given the difficulty score, the algorithm starts with a set of simple data points and gradually increases the dif-

**Figure 2. Examples for common data augmentation manipulations of images as presented by Taylor and Nitschke.[40]**


**Figure 3. An illustration of the learning setup used by Visotsky et al.[51]**

Labeled examples are accompanied with rich information that provides hints or explains classification. These labels are created during the training phase, where annotators write a list of objects, they observe in a visual scene using free text. Irrelevant background objects are often ignored. During the test phase, only the image is provided.

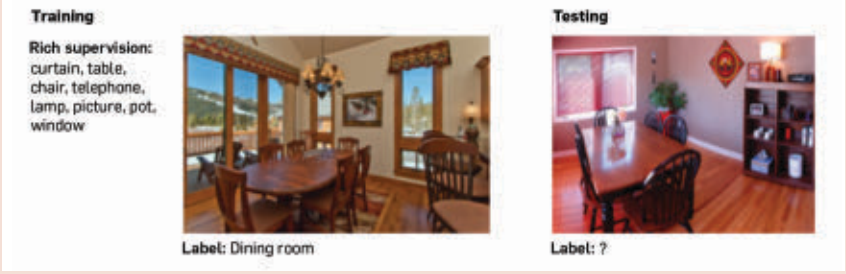
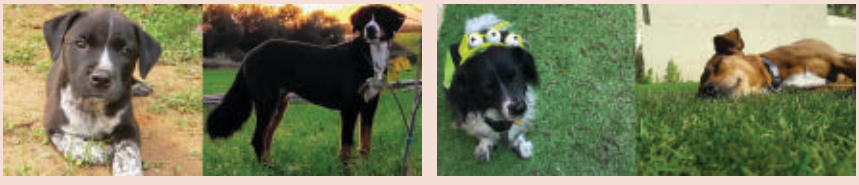



**Figure 4: Typical versus non typical images of dogs are considered to be easy versus hard, respectively, in a dogs versus cats classification task.**



**(a)** Typical image of dogs, easier to classify.

**(b)** Non-typical images of dogs, more difficult to classify.

ficulty of training examples throughout the learning process. This progression enables the model to learn the broad concept on a few easy examples and later refine the concept with more difficult ones. Figure 4 shows photos of dogs in the top row are more typical and should be easier for a classifier to recognize.

Curriculum learning has been shown to improve performance while decreasing the number of examples needed for convergence.[17] For example, Zaremba and Sutskever[58] showed how curriculum improves learning for the task of predicting the output of Python code without executing it.

A major caveat of curriculum learning is the inherent need for a difficulty-label estimator. Human labeling of difficulty can be very demanding, perhaps even more than standard annotation. In practice, the difficulty of each example is often learned by a teacher model, which may have access to related training data.[17]

A related concept is *self-paced learning* (SPL).[19] Intuitively, the curriculum in SPL is determined by the student's abilities, rather than being fixed by the teacher. Instead of heuristically designing a difficulty measure, SPL introduces a regularizor into the learning objective, with the goal of optimizing a curriculum for the model itself. This makes SPL broadly applicable.

### Argumentation-based machine learning

**Use experts' local knowledge to restrict the search space.**

*Argumentation-based machine learning* (ABML) is a method to constrain the search space using *experts' local knowledge*.[26] In a nutshell, in ABML the learner attempts to find if-then rules to explain argumented examples in a rule induction process. The learner starts by finding a rule, adding it to a set of rules and removing all training data points that are covered by that rule. This process is repeated until all examples are removed. ABML's main advantage is the use of expert knowledge to justify *specific* examples, which is often easier than explaining global phenomena.

For example, Možina et al.[26] used ABML for medical records of deceased patients, where they used a physician's reasoning for the cause of death to limit the search space.

ABML is perhaps less popular than the other methods in this section. Nevertheless, if expert local knowledge is available, ABML is a powerful way to integrate partial prior knowledge. Moreover, the induced hypothesis should make more sense to an expert, as it must be consistent with the input arguments.

**Models encoding relevant knowledge.** Here, we go beyond the classical pipeline of training a model for a task; we present models that can take advantage of other, related tasks.

**The data-bottleneck problem is widespread across the field of machine learning.**

### Multi-task learning

**Co-learn multiple tasks simultaneously to enhance cross task similarities for better generalization.**

*Multi-task learning* (MTL) is a prominent area of research where one attempts to train on multiple different (yet related) tasks simultaneously. These multiple tasks are solved concurrently, exploiting commonalities and differences across them.

It has been shown that challenging the learner to solve multiple problems at the same time results in better generalization and better performance on each individual task.[36] Indeed, MTL is successfully used in both vision and NLP. The key factors for this success in the absence of a large dataset are: It is an implicit data augmentation method, based on cross-task commonalities; it enables unraveling cross tasks and feature correlations; and encouraging a classifier to also perform well on a slightly different task is a better regularization than uninformed regularizers (for example, enforce weights to be small, which is the typical $L2$-regularization).

As an example, consider the case of spam-filtering. Quite often, data from an individual user is insufficient for training a model. Intuitively, different people have different distributions of features that distinguish spam from legitimate email. For example, email messages in Russian are probably spam for English speakers, but not for Russian speakers. However, inter-user commonalities can be utilized to solve this problem (for example, text related to money transfer is probably spam). To build upon these similarities, Attenberg et al.[4] created an MTL-based spam-filter, treating each in-

dividual user as one distinct but related classification task and training a model across the different users.

A more recent example of MTL learning is the T5 model (see Figure 5).[29] This model achieves state-of-the-art results on many NLP benchmarks while being flexible enough to be fine-tuned to a variety of downstream tasks. T5 receives as input the task at hand and thus allows the use of the same model, loss function, and hyperparameters for any NLP task.

MTL implementations can be divided into two main categories – hard versus soft parameter sharing of the hidden layers, where hard parameter sharing is more commonly used. In the hard type, the hidden layers are shared between all tasks while keeping several task-specific output layers. Baxter[5] showed that hard parameter sharing reduces the risk of overfitting to order N (the number of tasks), which is smaller than the risk of overfitting the task-specific parameters (the output layers). In soft parameter sharing, each task has its own model and parameters. The distance between model's parameters is then regularized to encourage them to be similar (enhance cross tasks' similarity), as done by Duong et al.[10]

## Transfer learning

**Transfer knowledge gained while solving one problem to a different yet related problem.**

*Transfer learning* is a widely used, highly effective way to integrate prior knowledge, like humans, who never approach a new problem tabula rasa, but rather with rich experience of somewhat similar problems and their solutions.[42]

The idea is to use preexisting models trained on related tasks. These pretrained models are usually used as an initialization for finetuning using a small dataset for the task in hand. Thus, significantly less task-specific examples are needed for convergence.

Another beneficial side effect is the use of the model's initial wide domain knowledge, compared to initialization with random weights. In other words, the model starts the fine-tuning phase with some relevant world knowledge.

For example, models trained on ImageNet have been transferred to medical imaging tasks, including inspecting chest x-rays[54] and retinal fundus images.[8] The idea is that a network trained on a large and diverse dataset of images captures universal visual features such as curves and edges in its early layers (similar to the primary visual cortex of humans and many other mammals, a Nobel prize winning discovery[a]). Despite the difference between the images in ImageNet and those in the downstream tasks, these features are relevant for many vision tasks. Therefore, this approach significantly decreases the size of labeled task-specific data needed.

In NLP, the commonly used pretrained model BERT achieves state-of-the-art results in various tasks.[9] Pretraining such models is often done in a *self-supervised* manner, where different parts of the input are masked, and the learner's goal is to predict the masked parts. For example, given a sentence, it is possible to iterate over it, masking a different word each time, to create various examples.

Fine-tuning in deep networks is usually done either by adding an untrained last layer and training the new model on the small task-specific dataset or by taking the output embeddings of the next to last layer. Another possible fine-tuning technique is to train the whole network with a relatively small learning rate; that is, perform small changes on the already-decent weights (as a heuristic, about 10 times smaller than the learning rate used for pretraining). Fine-tuning can also be done by freezing the weights of the first few layers of the pretrained model. The motivation behind this technique is that the first layers capture universal features that would probably also be relevant to the new task. Thus, freezing them during fine-tuning should keep the captured information that is relevant for both the original and the new tasks.

To conclude, transfer learning is a powerful tool for both reducing the amount of task-specific data needed and improving models' performance.

## Meta learning

**Improve the learning algorithm by generalizing based on experience from multiple learning episodes.**

*Meta-learning* (also known as "learning to learn") is a recent subfield of machine learning,[12] focusing on designing models that can learn new tasks or adapt to new environments rapidly, with only a few training examples. It is based on creating a meta-learner that has wide prior knowledge regarding the relevant topic(s). Meta learning is also inspired by human learning. For example, people who know how to ride a bicycle are more likely to quickly learn to ride a motorcycle.

Note that while meta learning can often be meaningfully combined with MTL systems, their objectives are different. While MTL aims to solve all training tasks, meta learning aims to use the training tasks for solving new tasks with small data. Thus, meta learning is about creating models with *prior experience* that can quickly adapt to new tasks. Specifically, the meta-learner gradually learns meta-knowledge across tasks, which can be

a https://www.nobelprize.org/uploads/2018/06/hubel-lecture.pdf

**Figure 5. Multi-task paradigm as presented by Raffel et al.[29]**

The objective is to train a model to perform several tasks that are closely related. The input contains the current task, which allows the use of same model, loss function and hyperparameters across various tasks.

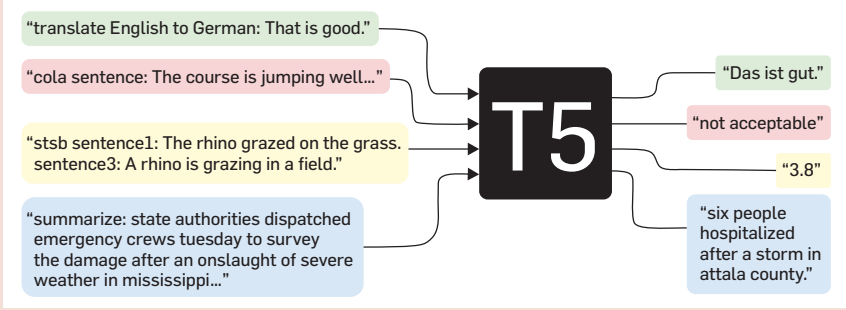


**Figure 6. Images generated by the GAN transformation approach for "near-miss" examples.[43]**

Images generated based on the MNIST dataset are interpretable to humans, while this is not the case for the CIFAR10 examples.

**(a)** MNIST **(b)** CIFAR 10

generalized to a new task using little task-specific information.

There are three common approaches to meta-learning: metric-based (similar to nearest-neighbor algorithms), optimization-based (meta-gradients optimizing), and model-based (no assumptions about data distribution).

As an example of a metric-based approach, Vinyals et al.[50] proposed a framework that explicitly learns from a given support set to minimize a loss over a batch. The result is a model that learns to map a small, labeled support set and an unlabeled example to its label, obviating the need for fine-tuning to adapt to new class types. They then showed the superiority of this method in both vision and NLP tasks.

A well-known work in the optimization-based line of research is model-agnostic meta-learning (MAML), which is a general optimization algorithm, compatible with any gradient descent-based model.[12] It uses a meta-loss specifically designed to induce quick changes when fine-tuned on new tasks and is based on *N*-gradients (where *N* is the total number of tasks).

In the model-based line of research, Munkhdalai and Yu[27] presented Meta-Net, a meta-learning model designed specifically for rapid generalization across tasks. The rapid generalization of MetaNet relies on "fast weights", which are parameters of the network with a smaller timescale for changes than the regular gradient-based weight changes. This Hebbian short-term plasticity maintains a dynamically changing short-term memory of the recent history of the units' activities in the network, as opposed to the standard slow recurrent connectivity. This model outperforms various other recurrent models across several tasks.

## Obstacle: Missing Labels

We now turn our attention to the second major branch in Figure 1, where unlabeled data is abundant, but there are few labels (or no labels at all). This setting is common in practice because unlabeled data is often much easier to obtain than labeled data. In this section we cover two main approaches. The first deals with ways to acquire labels efficiently, and the other uses weak labels.

### Active learning

**Generate examples which are close to the decision boundary. These examples should contribute to the learning process more than random examples.**

**Acquiring labels efficiently.** When more labels are needed but annotation is costly, an immediate question would be how to acquire new labeled data *efficiently*. The prime example of this is active learning, in which the learner can iteratively query an oracle (information source) to label new data points.[32] These queries can include unlabeled examples either from the dataset or new ex-nihilo data points, often ones that are close to the decision boundary. The rationale is that not all examples contribute equally to the learning process: diverse examples that are difficult for the learner to classify might be especially useful and could decrease the number of data points needed for learning.

There are many methods to determine which data points from the training set should be queried next. Common objectives include picking examples which will change the current model the most, examples which the current model is least certain about, or diverse examples that resemble the data distribution. For example, Hacohen et al.[16] recently showed that in the presence of little data it is most beneficial to present the model with typical examples (compared to scenarios with more data, in which it is best to use examples that are close to the decision boundary).

When generating *new* examples (rather than selecting unlabeled ones from the training set), it is important to remember that humans will be the ones labeling them. We wish to point out that while data augmentation modifies the input but *keeps* its label (as discussed earlier), active learning generates examples *without labels*. Thus, the generation algorithm should keep the new points interpretable, that is, ensure they have a clear label.[14] For example, Zarecki and Markovitch[57] automatically transformed sentences' sentiment by replacing key words that bring them closer to the classification boundary (while keeping their syntax).

Recent approaches use GANs to generate new examples, either from scratch (and label them),[60] or by modifying an existing example (while attempting to preserve the label).[43] Both scenarios update the learner and the GAN model simultaneously after labeling a new example.

Importantly, the GAN approaches are more expressive than transformation-based approaches, but the result is often less interpretable. Figure 6 shows an example of modified images from Tran et al.[43] Note that while the MNIST examples (handwritten digits) have relatively clear labels, the CIFAR10 examples (tiny images in ten classes such as airplane, dog, and ship) are not as easy to label.

*A note on gamification.* Active learning is the dominant paradigm for reducing the *number* of annotations needed. However, a different approach

to label efficiently is to reduce the cost of annotations. A notable example is *gamification*—applying gaming mechanics to non-gaming environments, to make tasks more enjoyable and give annotators a non-monetary incentive to provide labels. The challenge in gamification is often to design the game to create the right incentive. This is far from trivial, and requires knowledge of game design, motivational psychology, and an understanding of the target group.[25] Ignorance of the complexity involved in gamification often results in modest outcomes.

The seminal work of Von Ahn and Dabbish[52] demonstrated a two-player game for image labeling, where the players gain points for describing an image using the exact same term. The researchers famously estimated that if users were to play the game at the same rate as other popular online games, most images on the Web could be labeled (for free) within only a few months. Another example is the unfun.me corpus used in humor research. This corpus was constructed via an online game where players change satirical headlines into serious ones with minimal edits.[55]

**Weak labeling**. If we cannot obtain labels efficiently, we could choose to obtain noisy labels as a proxy. In vision, this is sometimes referred to as "automatic image annotation." We cover two main types of noisy labels here.

*Assumptions on* $P(Y=y|X=x)$.

## Semi-supervised learning

**Harness information regarding $P(X=x)$ to reduce labeling requirements by integrating labeled and non-labeled examples in the learning process.**

*Semi-supervised learning* (SSL) is a very large and active area of research, and we do not profess to cover all it; for a recent survey on SSL, we refer the reader to van Engelen and Hoos.[46]

SSL estimates the distribution $P(X=x)$ using a large amount of *unlabeled*, to reduce the annotated data requirements. It makes strong assumptions about the relation between $P(X=x)$ and $P(Y=y|X=x)$ to reduce the number of labeled examples needed.[56] Typically, these assumptions take the following forms:

▸ **Smoothness:** Points that are close to each other are more likely to share a label. More formally, every two adjacent samples $x, x'$ should have similar labels.

▸ **Cluster-ability:** Data tend to form discrete clusters where points belonging to the same cluster are more likely to share a label. Thus, the decision boundary can only pass through low-density areas in the feature space.

▸ **Manifold:** Data lies approximately on a manifold of a much lower dimension than the input space. Thus, when considering low-dimensional manifolds of the input space, any data points on the same manifold should have the same label.

All three assumptions can be seen as different definitions of interpoints similarity: The smoothness defines it as proximity in the input space, the cluster-ability assumes high-density areas contain similar data points, and the manifold states that points which lie on the same low-dimensional manifold are similar.

Another important distinction in SSL is between inductive and transductive methods. The former yields a classification model to predict the label of a new example, like supervised learning ($f: X \rightarrow Y$). The latter do not yield such a model, but instead directly provide predictions. Transductive approaches are usually graph-based, while the inductive approaches can be further divided into *unsupervised preprocessing*, *intrinsically semi-supervised*, and *wrapper* methods.[46]

One popular way of using the unsupervised preprocessing approach is to use the knowledge on $P(X=x)$ to extract useful features in a lower dimension than the original dimension of $X$ and thus reduce the learning complexity. This includes learning a representation using an auto-encoder model[49] or applying a dimensionality reduction method like PCA.[1]

Under the inductive approach, it is also possible to use an intrinsically semi-supervised model like semi-supervised SVM, which changes the optimization target to find a decision boundary with maximal margin from both labeled and unlabeled points (for example, using SVM).[47] This can also be applied to neural networks by adding a form of regularization over the unlabeled data.[30]

In wrapper methods, a model is initially trained from the available set of examples.[44] It then makes predictions on the unlabeled dataset. The model's

**When generating *new* examples (rather than selecting unlabeled ones from the training set), it is important to remember that humans will be the ones labeling them.**

pseudo-labels are added as labeled data for the next iteration of supervised learning. This process is repeated until convergence.

## Data programming

**Integrate multiple weak heuristics regarding the labeling process $f: X \rightarrow Y$ to create noisy labels.**

*Data programming* is a paradigm for the programmatic creation of training sets. In data programming, users express weak supervision strategies or domain heuristics as labeling functions (LFs), which are programs that label subsets of the data. Importantly, LFs are imprecise and can contradict each other, resulting in noisy labels. By explicitly representing the labeling process $f: X \rightarrow Y$ as a generative model, data programming aims to "denoise" the generated training set.

For example, in spam-detection, potential LFs would return "spam" if the email contains a URL or a money transfer request, and "no-spam" if coming from someone in your contact list. These functions alone achieve poor performance; however, like ensemble methods (where a group of weak learners comes together to form a strong one with superior accuracy), the strength of data programming is in the combination of many weak heuristics.

A popular system for data programming is Snorkel.[31] It applies the (noisy) LFs to the data and estimates their accuracy and correlations, using only their agreements and disagreements. This information is then used to reweight and combine LF predictions to output probabilistic noise-aware training labels. This process is presented in Figure 7.

## Expectation regularization

**Using prior knowledge regarding the proportion of the different labels in sub-groups of the data to create noisy labels.**

Prior knowledge regarding labels' *proportion* in various subgroups of the data, makes it possible to automatically create noisy labels in a process called *expectation regularization* (learn from label proportions).[53]

This estimation process relies on uniform convergence properties of the expectation operator. It uses empirical means of the sub-groups to approximate expectations with respect to a group's distribution. The latter is then used to compute expectations with respect to a given label, and finally, the conditional means on the label distribution are used to estimate the conditional group means.

A recent work in this area is *ballpark learning*, which relaxes the assumption of known label proportions, assuming instead soft constraints on proportions within and between groups of instances (for example, "the percentage of spam in emails mentioning a certain word is between $k_{low}$ and $k_{high}$", or "emails containing a link have at least $k$% more spam than emails without links").[18] Ballpark learning learns a model that labels individual instances while satisfying these soft, noisy constraints.

*Noisy Supervision from External Datasets.* It is sometimes possible to take advantage of preexisting datasets to get a noisy supervision signal.

## Distant supervision

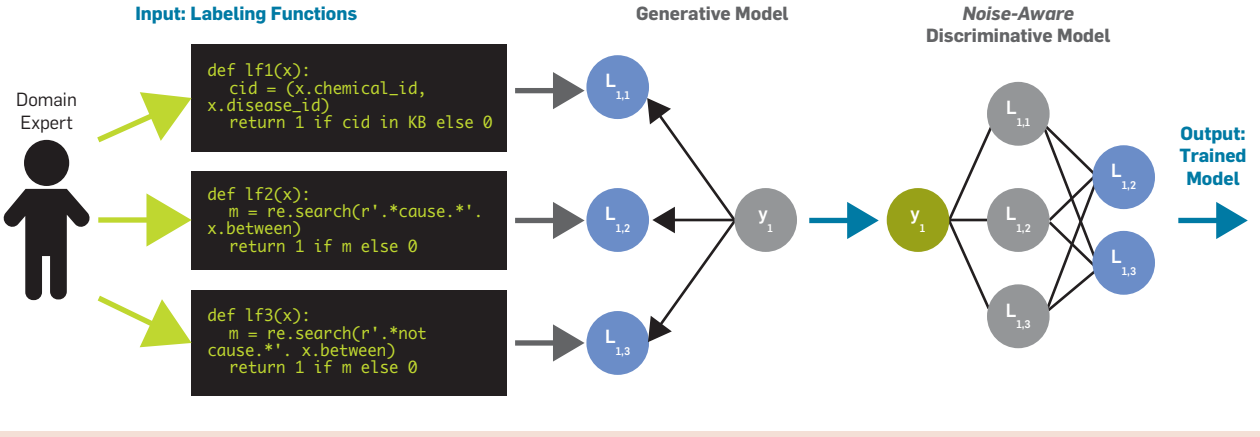
**Use a preexisting database to collect examples for the desired relation. These examples are then used to automatically generate labeled training data.**

*Distant supervision* is a popular method to use existing datasets. In distant supervision, a model is learned given a labeled training set, as in "standard" supervised ML, but the training data is weakly labeled (that is, labeled automatically, based on heuristics or rules).

For example, Mintz et al.[24] used Freebase, a large, unlabeled, semantic database, to provide distant supervision for relation extraction. The intuition is that any sentence that contains a pair of entities with a known Freebase relation is likely to express that relation in some way.[23] For example, each pair of "Barack Obama" and "Michelle Obama" that appear in the same sentence can be extracted as a positive example for the marriage relation. Due to the poten-

**Figure 7. Illustration of Snorkel's pipeline.**

The domain expert creates noisy labeling functions (LFs). A generative model learns to resolve and model the output of these LFs. The model's output is the input to a discriminative model. Image reproduced from https://towardsdatascience.com/snorkel-a-weak-supervision-system-a8943c9b639f



tially large number of sentences that contain a given entity pair, it is possible to extract and combine noisy features for the labeling process. Based on these semantic signals, Mintz et al.[24] was able to use 116 million unlabeled instances.

## Incidental supervision

**Exploit weak signals that exist in data independently of the task at hand.**

The *incidental supervision* framework is based on the idea that informative cues for a task could exist in datasets that were not constructed with this task in mind. For example, suppose we want to infer gender from first names. One could use Wikipedia, which was not created for this task. The incidental signal would be pronouns and other gender indicators appearing in the first paragraph of Wikipedia pages about people with that first name. This signal is correlated to the task at hand and (together with other signals and inferences), could be used for supervision, reducing the need for annotations.

Incidental supervision does not assume knowledge about the labeling process.[34] Moreover, incidental signals can be noisy, partial, or only weakly correlated with the target task, and still be used to provide supervision and facilitate learning. Note that the notion of supervision here is different from that of distant supervision: In distant supervision, the model learns in the standard supervised learning way, but the training set is labeled automatically, based on heuristics. In incidental supervision, a complete training set might never exist.

Context-sensitive spelling and grammar checking is a task that has been relying on incidental supervision for over 20 years now.[13] Under the assumption that most edited textual resources (books, newspapers, Wikipedia) do not contain many spelling and grammar errors, these methods generate contextual representations for words, punctuation marks and phenomena such as agreements. These representations are then used to identify mistakes and correct them in a context-sensitive manner.[35]

An unintentional example for the power of incidental signals comes from image processing, where the task of gender detection based on the iris texture was solved with great accuracy (over 80% for most papers and an impressive score of 99.5% reported by Alrashed and Berbar[2]). However, it was later discovered that most models did not detect a person's gender; rather, they detected the use of cosmetic mascara, which is a much easier task and is indeed correlated with the original assignment.[21] Thus, although unintentionally, this finding emphasized the potential of using incidental cues.

**Identifying assumptions is essential for breaking them—and breaking assumptions is an established technique for encouraging creativity and innovation.**

## Conclusion

The dominant paradigm in ML today is creating large, task-specific datasets (often using crowdsourcing). In this review we devise a taxonomy for alternative ways to tackle the data bottleneck problem. The taxonomy aims to bring order to the various methods suggested across different subfields, as well as making it easier to identify underlying assumptions and potential new directions. Identifying assumptions is essential for breaking them—and breaking assumptions is an established technique for encouraging creativity and innovation.

For example, surveying the taxonomy, several common assumptions that stand out are that samples tend to be representative of the data, that we have information about $X$ and $Y$ conjointly, and that each example has exactly one correct label. This raises the prospect of new learning settings (for example, what if we only have knowledge about the distributions of data points $P(X = x)$ and labels $P(Y = y)$, separately?), and of new ways to aggregate multiple (correct but different) labels.

We note that our taxonomy covers widely diverse techniques, making very different assumptions. Ultimately, we expect that choosing a technique will often boil down to what the practitioner has access to (that is, which assumptions are met). For example, in multitask learning the practitioner not only possess labeled data for their task, but also for several related tasks; in data programming, they have no (or very few) labels for their task but possess some partial knowledge about the labeling process; in curriculum learning, they know something about the hardness of data points; and so on.

We further wish to point out that it is not always obvious whether a method's assumptions are met in practice, or to estimate which method is better

more online
A version of this article with a comprehensive list of references is available at https://dx.doi.org/10.1145.3551635

suited for a specific use case. The answer might depend on many factors, such as the inherent difficulty of the concept one wishes to learn, biases in the data, or the manual effort needed to obtain high-quality input for the different methods. For example, in methods using weak labeling, the tradeoff between implementation speed and accuracy for different weak labels is often not clear in advance.

In addition to the inherent difficulty of collecting large datasets, we note there are growing concerns about such datasets, including environmental costs, financial costs, opportunity costs, and more.[38] We also note that large datasets are still prone to fitting artifacts, and that several recent methods have attempted to address the recurring challenges of the annotation artifacts and human biases found in many existing datasets.[15]

In conclusion, ML has made tremendous progress using large datasets, but they are not a panacea for all problems. Our hope is that this paper will encourage re-thinking about current annotation-heavy approaches.

**Acknowledgment.** This work was supported by a grant from Israel Ministry of Science and Technology and by the European Research Council (ERC) under European Union's Horizon 2020 research and innovation program (grant no. 852686, SIAM).

**References**


1. Alaíz, C., Fanuel, M., and Suykens, J. Convex formulation for kernel PCA and its use in semisupervised learning. *IEEE Trans. Neural Networks and Learning Systems 29*, 8 (2017), 3863–3869.
2. Alrashed, H. and Berbar, M. *Facial Gender Recognition Using Eyes Images* (2013).
3. Anaby-Tavor, A., et al. Do not have enough data? Deep learning to the rescue. In *Proceedings of the AAAI Conf. Artificial Intelligence 34* (2020), 7383–7390.
4. Attenberg, J., Weinberger, K., Dasgupta, A., Smola, A., and Zinkevich, M. Collaborative email-spam filtering with the hashing trick. CEAS (2009).
5. Baxter, J. A Bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning 28*, 1 (1997), 7–39.
6. Bertero, D. and Fung, P. Predicting humor response in dialogues from TV sitcoms. In *2016 IEEE Intern. Conf. Acoustics, Speech, and Signal Processing*, 5780–5784.
7. Cubuk, E., Zoph, B., Mané, D., Vasudevan, V., and Le, Q. AutoAugment: Learning augmentation strategies from data. In *Proceedings of 2019 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, 113–123.
8. De Fauw, J., et al. Clinically applicable deep learning for diagnosis and referral in retinal disease. *Nature Medicine 24*, 9 (2018), 1342–1350.
9. Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conf. the North American Chapter of the Assoc. Computational Linguistics: Human Language Technologies, 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, MN, 4171–4186. https://doi.org/10.18653/v1/N19-1423
10. Duong, L., Cohn, T., Bird, S., and Cook, P. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Assoc. Computational Linguistics* and the *7th Intern. Joint Conf. Natural Language Processing 2* (Short Papers, 2015, 845–850.
11. Farhadi, A., Endres, I., Hoiem, D., and Forsyth, D. Describing objects by their attributes. In *Proceedings of the 2009 IEEE Conf. Computer Vision and Pattern Recognition*, 1778–1785.
12. Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th Intern. Conf. Machine Learning-Volume 70*. JMLR. Org, 2017, 1126–1135.
13. Golding, A. and Roth, D. A winnow-based approach to context-sensitive spelling correction. *Machine Learning 34*, 1–3 (1999), 107–130.
14. Gurevich, N., Markovitch, S., and Rivlin, E. Active Learning with near Misses. AAAI Press, 2006, 362–367.
15. Gururangan, S., Swayamdipta, S., Levy, O., Schwartz, R., Bowman, S., and Smith, N. Annotation artifacts in natural language inference data. 2018; *arXiv:1803.02324*.
16. Hacohen, G., Dekel, A., and Weinshall, D. Active learning on a budget: Opposite strategies suit high and low budgets. 2022; *arXiv:2202.02794*.
17. Hacohen, G. and Weinshall, D. On the power of curriculum learning in training deep networks. 2019; arXiv:1904.03626.
18. Hope, T. and Shahaf, D. Ballpark learning: Estimating labels from rough group comparisons. In *Proceedings of the Joint European Conf. Machine Learning and Knowledge Discovery in Databases*. Springer, 2016, 299–314.
19. Jiang, L., Meng, D., Zhao, Q., Shan, S., and Hauptmann, A. Self-paced curriculum learning. In *Proceedings of the 29th AAAI Conf. Artificial Intelligence*, 2015.
20. Kottur, S., Vedantam, R., Moura, J., and Parikh, D. VisualWord2Vec (Vis-W2V): Learning visually grounded word embeddings using abstract scenes. In *Proceedings of the 2016 IEEE Conf. Computer Vision and Pattern Recognition*, 2015, 4985–4994.
21. Kuehlkamp, A., Becker, B., and Bowyer, K. Gender-from-iris or gender-from-mascara. In *Proceedings of the 2017 IEEE Winter Conf. Applications of Computer Vision*, 1151–1159.
22. Lampert, C., Nickisch, H., and Harmeling, S. Learning to detect unseen object classes by between-class attribute transfer. In *Proceedings of the 2009 IEEE Conf. Computer Vision and Pattern Recognition*, 951–958.
23. Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the 29th AAAI Conf. Artificial Intelligence*, 2015.
24. Mintz, M., Bills, S., Snow, R., and Jurafsky, D. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conf. 47th Annual Meeting of the ACL* and the *4th Intern. Joint Conf. Natural Language Processing of the AFNLP, 2*. Assoc. Computational Linguistics, 2009, 1003–1011.
25. Morschheuser, B. and Hamari, J. The gamification of work: Lessons from crowdsourcing. *J. Management Inquiry 28*, 2 (2019), 145– 148.
26. Možina, M., Žabkar, J., and Bratko, I. Argument based machine learning. *Artificial Intelligence 171*, 10-15 (2007), 922–937.
27. Munkhdalai, T. and Yu, H. Meta networks. In *Proceedings of the 34th Intern. Conf. Machine Learning 70*. JMLR. Org, 2017, 2554–2563.
28. Perez, L. and Wang, J. The effectiveness of data augmentation in image classification using deep learning. 2017; *arXiv:1712.04621*.
29. Raffel, C., et al. Exploring the limits of transfer learning with a unified text-to-text transformer. 2019; *arXiv:1910.10683*.
30. Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. Semi-supervised Learning with Ladder Networks. In *NIPS*, 2015.
31. Ratner, A., Bach, S., Ehrenberg, H., Fries, J., Wu, S., and Ré, C. Snorkel: Rapid training data creation with weak supervision. *The VLDB J.* (2019), 1–22.
32. Ren, P., et al. A survey of deep active learning. *ACM Computing Surveys 54*, 9 (2021), 1–40.
33. Roh, Y., Heo, G., and Whang, S. A survey on data collection for machine learning: a big data-ai integration perspective. *IEEE Trans. Knowledge and Data Engineering 33*, 4 (2019), 1328–1347.
34. Roth, D. Incidental supervision: Moving beyond supervised learning. In *Proceedings of the 31st AAAI Conf. Artificial Intelligence*, 2017.
35. Rozovskaya, A. and Roth, D. Building a state-of-the-art grammatical error correction system. *Tran. Assoc. Computational Linguistics 2* (2014), 419–434.
36. Ruder, S. An Overview of Multi-Task Learning in Deep Neural Networks. 2017; *arXiv abs/1706.05098* (2017).
37. Schwartz, E., Karlinsky, L., Feris, R., Giryes, R., and Bronstein, A. Baby steps towards few-shot learning with multiple semantics. arXiv preprint *arXiv:1906.01905* (2019).
38. Schwartz, R., Dodge, J., Smith, N., and Etzioni, O. Green AI. *arXiv preprint arXiv:1907.10597*, 2019.
39. Shin, Y., Sagong, M., Yeo, Y., Kim, S., and Ko, S. Pepsi++: fast and lightweight network for image inpainting. *IEEE Trans. Neural Networks and Learning Systems* (2020).
40. Taylor, L. and Nitschke, G. Improving deep learning using generic data augmentation. 2017; *arXiv:1708.06020*.
41. Tian, Y., Shi, J., Li, B., Duan, Z., and Xu, C. Audiovisual event localization in unconstrained videos. In *Proceedings of the 2018 European Conf. Computer Vision*, 247–263.
42. Torrey, L. and Shavlik, J. Transfer learning. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. IGI Global, 2010, 242–264.
43. Tran, T., Do, T., Reid, I., and Carneiro, G. Bayesian Generative Active Deep Learning. 2019; *arXiv abs/1904.11643*.
44. Triguero, I., García, S., and Herrera, F. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information Systems 42* (2013), 245–284.
45. van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel Recurrent Neural Networks. In *ICML, 2016*.
46. van Engelen, J. and Hoos, H. A survey on semi-supervised learning. *Machine Learning* (2019), 1–68.
47. Vapnik, V. Statistical learning theory (1998).
48. Vapnik, V. The vicinal risk minimization principle and the SVMs. T*he Nature of Statistical Learning Theory*. Springer, 2000, 267–290.
49. Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P. Extracting and composing robust features with denoising autoencoders. *ICML '08*.
50. Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. *Advances in Neural Information Processing Systems 29* (2016), 3630–3638.
51. Visotsky, R., Atzmon, Y., and Chechik, G. Few-shot learning with per-sample rich supervision. 2019; *arXiv:1906.03859* (2019).
52. Von Ahn, L. and Dabbish, L. Labeling images with a computer game. In *Proceedings of the 2004 SIGCHI Conf. Human Factors in Computing Systems*, 319–326.
53. Wang, M. and. Manning, C. Cross-lingual projected expectation regularization for weakly supervised learning. *Trans. Assoc. Computational Linguistics 2* (2014), 55–66; https://doi.org/10.1162/ tacl_a_00165
54. Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., and Summers, R. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the 2017 IEEE Conf. Computer Vision and Pattern Recognition*, 2097–2106.
55. West, R. and Horvitz, E. Reverse-engineering satire, or 'paper on computational humor accepted despite making serious advances.' In *Proceedings of the 2019 AAAI Conf. Artificial Intelligence 33*, 7265–7272.
56. Xiaojin, Z. Semi-supervised learning literature survey. *Computer Sciences TR 1530* (2008).
57. Zarecki, J. and Markovitch, S. Textual membership queries. In *Proceedings of the 29th Intern. Conf. Intern. Joint Conf. Artificial Intelligence*, 2021, 2662–2668.
58. Zaremba, W. and Sutskever, I. Learning to execute. 2014; *arXiv:1410.4615*.
59. Zhang, H., Cissé, M., Dauphin, Y., and Lopez-Paz, D. Mixup: Beyond Empirical Risk Minimization. 2017; *arXiv abs/1710.09412*.
60. Zhu, J. and Bento, J. Generative Adversarial Active Learning. 2017; *arXiv abs/1702.07956*.


**Chen Shani** is a Ph.D. student at The Hebrew University of Jerusalem, Israel.

**Jonathan Zarecki** is R&D Group Lead for Israeli Military Intelligence, Tel Aviv, Israel.

**Dafna Shahaf** is an associate professor of data science at The Hebrew University of Jerusalem, Israel.

DOI:10.1145/3576893

To view the accompanying paper, visit doi.acm.org/10.1145/3576894 rh

# Technical Perspective
# Beautiful Symbolic Abstractions for Safe and Secure Machine Learning

By Martin Vechev

OVER THE LAST decade, machine learning has revolutionized entire areas of science ranging from drug discovery to autonomous driving, to medical diagnostics, to natural language processing and many others. Despite this impressive progress, it has become increasingly evident that modern machine learning models suffer from several issues which, if not resolved, could prevent their widespread adoption. Example challenges include lack of robustness guarantees to slight distribution shifts, reinforcing unfair bias present in training data, leakage of sensitive information through the model, and others.

Addressing these issues by inventing new methods and tools for establishing that machine learning models enjoy certain desirable guarantees, is critical, especially for domains where safety and security are paramount. Indeed, over the last few years there has been substantial research progress in new techniques aiming to address the above issues with most work so far focusing on perturbations applied to inputs of the model. For instance, the community has developed novel verification methods for proving that a model always classifies a sample (for example, an image) to the same label regardless of certain transformations (for example, an arbitrary rotation of up to five degrees). New sophisticated methods are constantly being invented targeting different properties, different types of guarantees (probabilistic, deterministic) and application domains (for example, natural language or visual perception).

Despite remarkable progress, there has been comparatively little work on proving that models work as intended when their *training data* has been manipulated (as opposed to manipulating test- or inference-time inputs). Addressing this challenge is important, as otherwise an adversary can poison the training data by inserting or modifying samples, to force the model into specific mis-predictions, leading to degraded performance at inference-time. This predicament can easily occur in various settings, for instance, in scenarios where models rely on large amounts of public (now potentially poisoned) data for their pretraining phase. At the same time, the problem is inherently difficult: proving that standard test-time performance is robust to changes of the training data requires reasoning about a potentially intractable number of models, each induced by a different subset of training data points. This means that a naive solution where one simply trains different models for different subsets of the data and then runs the test set through each of these models, is practically infeasible and raises the following fundamental question: how do we process an intractably large set of trained models when enumeration is infeasible?

The following PLDI paper addresses this challenging question in a clean, beautiful, and elegant manner. The key observation is to connect ideas from abstract interpretation with machine learning, by representing the intractably large set of trained models via a symbolic abstraction and introducing what amounts to a symbolic learning algorithm: rather than training a concrete model on a concrete dataset, the paper shows how to train a symbolic model (which captures an intractable number of concrete models) on a symbolic dataset (which captures an intractable number of possible concrete dataset variants). Creating the novel symbolic learning algorithm is non-trivial and requires introducing symbolic (abstract) counterparts for all key steps of the learning algorithm (for example, those computing outcome probabilities) and capturing their effect on the symbolic model. A key idea here is to train one symbolic model for each point in the test set, enabling a more scalable construction than a procedure which would be agnostic to the test set. Finally, given the symbolic model for that point, one tries to prove, via standard symbolic methods, that the point is indeed classified to the same label by that model.

Naturally, because the used abstractions are incomplete (to ensure scalability), this final proof step may fail even though the property holds. Importantly, however, the technique is sound: if the method states that a given point is classified to a particular label under the symbolic model, then the statement is guaranteed to hold. To demonstrate the overall concept, the paper focuses on a popular class of machine learning models, namely decision trees, but the introduced ideas are of general applicability.

The paper is a beautiful reference and a must-study for anyone interested in learning how to fruitfully connect, in a clean and elegant manner, the seemingly disparate worlds of data-driven optimization and symbolic techniques. Indeed, the creative fusion of these two fields will likely drive many new advances in next-generation AI systems. As to future work, now that we know progress is feasible, one would expect the paper to trigger much interest in proving properties involving training set perturbations, as well as in extending and applying these ideas to other kinds of models and application domains. This would necessarily require novel abstractions, resulting in new kinds of symbolic learning algorithms. C

**Martin Vechev** is a professor at ETH Zurich, Switzerland, where he leads the Secure, Reliable, and Intelligent Systems Lab.

DOI:10.1145/3576894

To view the accompanying Technical Perspective, visit doi.acm.org/10.1145/3576893

# Proving Data-Poisoning Robustness in Decision Trees

By Samuel Drews, Aws Albarghouthi, and Loris D'Antoni

**Abstract**

**Machine learning models are brittle, and small changes in the training data can result in different predictions. We study the problem of proving that a prediction is robust to *data poisoning*, where an attacker can inject a number of malicious elements into the training set to influence the learned model. We target decision tree models, a popular and simple class of machine learning models that underlies many complex learning techniques. We present a sound verification technique based on *abstract interpretation* and implement it in a tool called Antidote. Antidote abstractly trains decision trees for an intractably large space of possible poisoned datasets. Due to the soundness of our abstraction, Antidote can produce proofs that, for a given input, the corresponding prediction would not have changed had the training set been tampered with or not. We demonstrate the effectiveness of Antidote on a number of popular datasets.**

## 1. INTRODUCTION

Artificial intelligence, in the form of machine learning (ML), is rapidly transforming the world as we know it. Today, ML is responsible for an ever-growing spectrum of sensitive decisions—from loan decisions, to diagnosing diseases, to autonomous driving. Many recent works have shown how ML models are brittle,[2,5,19,20,22] and with ML spreading across many industries, the issue of robustness in ML models has taken center stage. The research field that deals with studying robustness of ML models is referred to as *adversarial machine learning*. In this field, researchers have proposed many definitions that try to capture robustness to different *adversaries*. The majority of these works have focused on verifying or improving the model's robustness to *test-time attacks*,[8,9,21] where an adversary can craft small perturbations to input examples that fool the ML model into changing its prediction, for example, making the model think a picture of a cat is that of a zebra.[4]

**Data-poisoning robustness.** This paper focuses on verifying *data-poisoning robustness*, which captures how robust a training algorithm $L$ is to variations in a given *training set* $T$. Intuitively, applying $L$ to the training set $T$ results in a classifier (model) $M$, and in this paper, we are interested in how the trained model varies when producing perturbations of the input training set $T$.

The idea is that an adversary can produce slight modifications of the training set, for example, by supplying a small amount of malicious training points, to influence the produced model and its predictions. This attack model is possible when the data is curated, for example, *via* crowdsourcing or from online repositories, where attackers can try to add malicious elements to the training data. For instance, one might add malicious training points to affect a malware detection model,[23] or a small number of images to bypass a facial recognition model.[5]

**Verification challenges.** Data-poisoning robustness has been studied extensively.[2,13,15,24,25] While some defenses have been proposed against specific attacks[10,19]—that is, modifications to training sets—we are not aware of any technique that can formally verify that a given learning algorithm is robust to perturbations to a given training set. Verifying data-poisoning robustness of a given learner requires solving a number of challenges. First, the datasets over which the learner operates can have thousands of elements, making the number of modified training sets we need to consider intractably large to represent and explore explicitly. Second, because learners are complicated programs that employ complex metrics (e.g., entropy and loss functions), their verification requires new techniques.

**Our approach.** We present a new *abstraction-interpretation-based approach* on the problem of verifying data-poisoning robustness for *decision tree learners*. We choose decision trees because they are simple, widely used, interpretable models. We summarize our contributions as follows:

- Antidote: the first sound technique for verifying data-poisoning robustness for decision tree learners (§2).
- A *trace-based view* of decision tree learning as a stand-alone algorithm that allows us to sidestep the challenging problem of reasoning about the set of all possible output trees a learner can output on different datasets (§3).
- An *abstract domain that concisely encodes sets of perturbed datasets* and the abstract transformers necessary to verify robustness of a decision tree learner (§4).
- An evaluation of Antidote on five representative datasets from the literature. Antidote can prove poisoning robustness for all datasets in cases where an enumeration approach would be doomed to fail (§5).

The original version of this paper was published in the *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, June 20, 2020.

**Figure 1. High-level overview of our approach.**

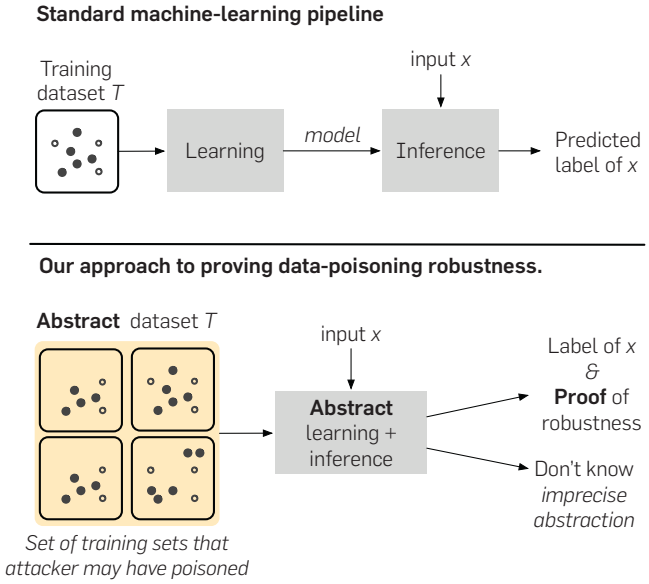
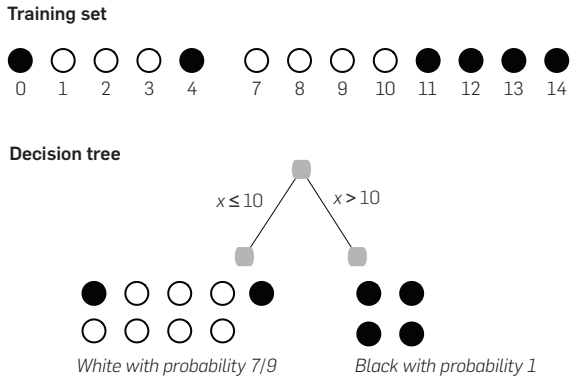



**Figure 2. Illustrative example.**




## 2. OVERVIEW

In this section, we give an overview of decision tree learning, the poisoning robustness problem, and motivate our abstraction-based proof technique (Figure 1).

**Decision tree learning.** Consider the dataset $T_{bw}$ at the top of Figure 2. It is comprised of 13 elements with a single numerical feature. Each element is labeled as a white (empty) or black (solid) circle. We use $x$ to denote the feature value of each element. Our goal is to construct a decision tree that classifies a given number into white or black.

For simplicity, we assume that we can only build trees of depth 1, like the one shown at the bottom Figure 2. At each step of building a decision tree, the learning algorithm is looking for a predicate $\varphi$ with the best score, with the goal of splitting the dataset into two pieces with *least diversity*; that is, most elements have the same class (formally defined usually using a notion of entropy). This is what we see in our example: Using the predicate $x \leq 10$, we split the dataset into two sets, one that is mostly white (left) and one that is completely black (right). This is the best split we can have for our data, assuming we can only pick predicates of the form $x \leq c$, for an integer $c$. Note that, while the set of predicates $x \leq c$ is infinite, for this dataset (and in general for any dataset), there exists only finitely many inequivalent predicates—for example, $x \leq 4$ and $x \leq 5$ split the dataset into the same two sets.

Given a new element for a classification, we check if it is $\leq 10$, in which case we say it is white with probability 7/9—that is, the fraction of white elements such that $\leq 10$. Otherwise, if the element is $> 10$, we say it is black with probability 1.

**Data-poisoning robustness.** Imagine we want to classify an input $x$ but want to make sure the classification would not have changed had the training data been slightly different. For example, maybe some percentage of the data was maliciously added by an attacker to sway the learning algorithm, a problem known as *data poisoning*. Our goal is to check whether the classification of $x$ is robust to data poisoning.

**A naïve approach.** Consider our running example and imagine we want to classify the number 5. Additionally, we want to prove that *removing up to two elements* from the training set would not change the classification of 5—that is, we assume that up to ∼15% (or 2/13) of the dataset is contributed maliciously. The naïve way to do this is to consider every possible training dataset with up to two elements removed and retrain the decision tree. If all trees classify the input 5 as white, the classification is robust to this level of poisoning.

Unfortunately, this approach is intractable. Even for our tiny example, we have to train 92 trees $\left(\binom{13}{2} + \binom{13}{1} + 1\right)$. For a dataset of 1,000 elements and a poisoning of up to 10 elements, we have $\sim 10^{23}$ possibilities.

**An abstract approach.** Our approach to efficiently proving poisoning robustness exploits a number of insights. First, we can perform decision tree learning *abstractly* on a *symbolic set of training sets*, without having to deal with a combinatorial explosion. The idea is that the operations in decision tree learning, for example, selecting a predicate and splitting the dataset, do not need to look at every concrete element of a dataset, but at aggregate statistics (counts).

Recall our running example in Figure 2. Let us say that up to two elements have been removed. No matter what two elements you choose, the predicate $x \leq 10$ remains one that gives *a* best split for the dataset. In cases of ties between predicates, our algorithm abstractly represents all possible splits. For each predicate, we can symbolically compute best- and worst-case scores in the presence of poisoning as an *interval*. Similarly, we can also compute an interval that overapproximates the set of possible classification probabilities. For instance, in the left branch of the decision tree, the probability will be [0.71, 1] instead of 0.78 (or 7/9). The best-case probability of 1 is when we drop the black points 0 and 4; the worst-case probability of 0.71 (or 5/7) is when we drop any two white points.

The next insight that enables our approach is that we *do not need to explicitly build the tree*. Since our goal is to prove robustness of a single input point, which effectively takes a single trace through the tree, we mainly need to keep track of the abstract training sets as they propagate along those

traces. This insight drastically simplifies our approach; otherwise, we would need to somehow abstractly represent sets of elements of a tree data structure, a nontrivial problem in program analysis.

**Abstraction and imprecision.** We note that our approach is sound but necessarily incomplete; that is, when our approach returns "robust" the answer is correct, but there are robust instances for which our approach will not be able to prove robustness. There are numerous sources of imprecision due to overapproximation, for example, we use the *interval domain* to capture real-valued entropy calculations of different training set splits, as well as the final probability of classification.

**An involved example.** To further illustrate our technique, we preview one of our experiments. We applied our approach to the MNIST-1-7 dataset, which has been used to study data poisoning for deep neural networks[19] and support vector machines.[2] In our experiments, we checked whether Antidote could prove data poisoning for the inputs used in the same dataset when training a decision tree. For example, when applying Antidote to the image of the digit in Figure 3, Antidote proves that it is poisoning robust (always classified as a seven) for up to 192 poisoned elements in 90 seconds. This is equivalent to training on $\sim 10^{432}$ datasets!

## 3. PRELIMINARIES

In this section, we begin by formally defining the *data-poisoning-robustness problem*. Then, we present a *trace-based* view of decision tree learning, which will pave the way for a poisoning robustness proof technique.

### 3.1. The poisoning robustness problem

In a typical supervised learning setting, we are given a learning algorithm $L$ and a training set $T \subseteq \mathcal{X} \cdot \mathcal{Y}$ comprised of elements of some set $\mathcal{X}$, each with its classification label from a finite set of classes $\mathcal{Y}$. Applying $L$ to $T$ results in a classifier (or model): $M : \mathcal{X} \to \mathcal{Y}$. For now, we assume that both the learning algorithm $L$ and the models it learns are deterministic functions.

A *perturbed set* $\Delta(T) \subseteq 2^{\mathcal{X} \times \mathcal{Y}}$ defines a set of possible *neighboring* datasets of $T$. Our robustness definitions are relative to some given perturbation $\Delta$. (In Section 4.1, we define a specific perturbed set that captures a particular form of data poisoning.)

Definition 3.1 (Poisoning Robustness). *Fix a learning algorithm $L$, a training set $T$, and let $\Delta(T)$ be a perturbed set. Given an element $x \in \mathcal{X}$, we say that $x$ is robust to poisoning $T$ if and only if*

$$\forall T' \in \Delta(T). \quad L(T')(x) = L(T)(x)$$

*When $T$ and $\Delta$ are clear from context, we will simply say that $x$ is robust.*

In other words, no matter what dataset $T' \in \Delta(T)$ we use to construct the model $M = L(T')$, we want $M$ to always return the same classification for $x$. returned for $x$ by the model $L(T)$ learned on the original training set $T$.

Example 3.1. *Imagine we suspect that an attacker has contributed 10 training points to T, but we do not know which ones. We can define $\Delta(T)$ to be T as well as every subset of T of size $|T| - 10$. If an input x is robust for this definition of $\Delta(T)$, then no matter whether the attacker has contributed 10 training items or not, the classification of x does not change.*

### 3.2. Decision trees: a trace-bas ed view

We will formalize a tree as the *set of traces* from the root to each of the leaves. As we will see, this trace-based view will help enable our proof technique.

A decision tree $R$ is a finite set of traces, where each trace is a tuple $(\sigma, y)$ such that $\sigma$ is a sequence of Boolean predicates and $y \in \mathcal{Y}$ is the classification.

Semantically, a tree $R$ is a function in $\mathcal{X} \to \mathcal{Y}$. Given an input $x \in \mathcal{X}$, applying $R(x)$ results in a classification $y$ from the trace $(\sigma, y) \in R$ where $x$ satisfies all the predicates in the sequence $\sigma = [\varphi_1, ..., \varphi_n]$, that is, $\wedge_{i=1}^{n} x \vDash \varphi_i$ is true. We say a tree $R$ is *well-formed* if for every $x \in \mathcal{X}$, there exists exactly one trace $(\sigma, y) \in R$ such that $x$ satisfies all predicates in $\sigma$. In the following, we assume all trees are well-formed.

Example 3.2 (Decision tree traces). *Consider the decision tree in Figure 2. It contains two traces, each with a sequence of predicates containing a single predicate:* $([x \leq 10]$, *white*$)$ *and* $([x > 10]$, *black*$)$.

### 3.3. Tree learning: a trace-based view

We now present a simple decision tree learning algorithm, Trace. Then, in Section 4, we abstractly interpret Trace with the goal of proving poisoning robustness.

One of our key insights is that we do not need to explicitly represent the learned trees (i.e., the set of all traces), since our goal is to prove robustness of a *single input* point, which effectively takes a *single trace* through the tree. Therefore, in this section, we will define a *trace-based decision tree learning algorithm*. This is inspired by standard algorithms—such as CART,[3] ID3,[16] and C4.5[17]—but *it is input-directed, in the sense that it only builds the trace of the tree that a given input x will actually traverse.*

**A trace-based learner.** Our trace-based learner Trace is shown in Figure 4. It takes a training set $T$ and an input $x$ and computes the trace traversed by $x$ in the tree learned on $T$. Intuitively, if we compute the set of all traces Trace$(T, x)$ for each $x \in T$, we get the full tree, the one that we would have traditionally learned for $T$.

The learner Trace repeats two core operations: $(i)$ selecting a predicate $\varphi$ with which to split the dataset (using bestSplit) and $(ii)$ removing elements of the training set based on whether they satisfy the predicate $\varphi$ (depending on $x$, using

**Figure 3. Example MNIST-1-7 digit that is proven poisoning-robust by Antidote.**

filter). The number of times the loop is repeated ($d$) is the maximum depth of the trace that is constructed. Throughout, we assume a fixed set of classes $\mathcal{Y} = \{1, ..., k\}$.

The mutable state of Trace is the triple $(T, \varphi, \sigma)$: (*i*) $T$ is the training set, which will keep getting refined (by dropping elements) as the trace is constructed. (*ii*) $\varphi$ is the most recent predicate along the trace, which is initially undefined (denoted by $\Diamond$). (*iii*) $\sigma$ is the sequence of predicates along the trace, which is initially empty.

**Predicate selection.** We assume that Trace is equipped with a finite set of predicates $\Phi$ with which it can construct a decision tree classifier; each predicate in $\Phi$ is a Boolean function in $\mathcal{X} \to \mathbb{B}$.

bestSplit($T$) computes a predicate $\varphi^* \in \Phi$ that splits the current dataset $T$—usually minimizing a notion of entropy. Ideally, the learning algorithm would consider every possible sequence of predicates to partition a dataset in order to arrive at an optimal classifier. For efficiency, a decision tree-learning algorithm does this greedily: It selects the best predicate it can find for a single split and moves on to the next split. To perform this greedy choice, it measures how diverse the two datasets resulting from the split are. We formalize this below:

We use $T\downarrow_\varphi$ to denote the subset of $T$ that satisfies $\varphi$, that is,

$$T\downarrow_\varphi = \{(x, y) \in T \mid x \vDash \varphi\}$$

Let $\Phi'$ be the set of all predicates that do not trivially split the dataset: $\Phi' = \{\varphi \in \Phi \mid T\downarrow_\varphi \neq \emptyset \wedge T\downarrow_{\neg\varphi} \neq T\}$. Finally, bestSplit($T$) is defined as follows:

$$\text{bestSplit}(T) = \arg\min_{\varphi \in \Phi'} \text{score}(T, \varphi)$$

where score($T$, $\varphi$) = $|T\downarrow_\varphi| \cdot \text{ent}(T\downarrow_\varphi) + |T\downarrow_{\neg\varphi}| \cdot \text{ent}(T\downarrow_{\neg\varphi})$. Informally, bestSplit($T$) is the predicate that splits $T$ into two sets with the lowest entropy, as defined by the function ent in Figure 5. Formally, ent computes *Gini impurity*, which is used, for instance, in the CART algorithm.[3] Note that if $\Phi' = \emptyset$, we assume bestSplit($T$) is undefined (returns $\Diamond$). If multiple predicates are possible values of bestSplit($T$), we assume one is returned nondeterministically. Later, in Section 4, our abstract interpretation of Trace will actually capture all possible predicates in the case of a tie.

**Figure 4. Trace-based decision tree learner Trace.**

**Input:** training set $T$ and input $x \in \mathcal{X}$
**Initialize:** $\varphi \leftarrow \Diamond$, $\sigma \leftarrow$ empty trace
repeat $d$ times
  **if** ent($T$) = 0 **then** return
  $\varphi \leftarrow$ bestSplit($T$)
  **if** $\varphi = \Diamond$ **then** return
  $T \leftarrow$ filter($T, \varphi, x$)
  **if** $x \vDash \varphi$ **then** $\sigma \leftarrow \sigma\varphi$ **else** $\sigma \leftarrow \sigma\neg\varphi$
**Output:** argmax$_{i \in [1,k]} p_i$, where cprob($T$) = $\langle p_1, \ldots, p_k \rangle$

**Figure 5. Auxiliary operator definitions. ent is Gini impurity; cprob returns a vector of classification probabilities, one element for each class $i \in [1, k]$.**

$$\text{ent}(T) = \sum_{i=1}^{k} p_i (1 - p_i), \text{ where } \text{cprob}(T) = \langle p_1, \ldots, p_k \rangle$$

$$\text{cprob}(T) = \left\langle \frac{|\{(x, y) \in T \mid y = i\}|}{|T|} \right\rangle_{i \in [1,k]}$$

Example 3.3. *Recall our example from Section 2 and Figure 2. For readability, we use $T$ instead of $T_{bw}$ for the dataset. Let us compute* score($T$, $\varphi$), *where $\varphi$ is $x \leq 10$. We have $|T\downarrow_\varphi| = 9$ and $|T\downarrow_{\neg\varphi}| = 4$. For the classification probabilities, defined by* cprob *(Figure 5), we have* cprob($T\downarrow_\varphi$) = $\langle 7/9, 2/9 \rangle$ *and* cprob($T\downarrow_{\neg\varphi}$) = $\langle 0, 1 \rangle$ *assuming the first element represents white classification; for example, in $T\downarrow_\varphi$, there's a 7/9 chance of being classified as white. For* ent, *we have* ent($T\downarrow_\varphi$) ≈ 0.35 *and* ent($T\downarrow_{\neg\varphi}$) = 0. *Since $T\downarrow_\varphi$ only comprises black points, its Gini impurity is 0.*

*The score of $x \leq 10$ is therefore* ∼3.1. *For the predicate $x \leq 11$, we get the higher (worse) score of* ∼3.2, *as it generates a more diverse split.*

**Filtering the dataset.** The operator filter removes elements of $T$ that evaluate differently than $x$ on $\varphi$. Formally,

$$\text{filter}(T, \varphi, x) = \begin{cases} T\downarrow_\varphi & \text{if } x \vDash \varphi \\ T\downarrow_{\neg\varphi} & \text{otherwise} \end{cases}$$

**Learner result.** When Trace terminates in a state $(T_r, \varphi_r, \sigma_r)$, we can read the classification of $x$ as the class $i$ with the highest number of training elements in $T_r$.

Using cprob, in Figure 5, we compute the probability of each class $i$ for a training set $T$ as a vector of probabilities. Trace returns the class with the highest probability:

$$\arg\max_{i \in [1,k]} p_i \qquad \text{where cprob}(T_r) = \langle p_1, \ldots, p_k \rangle$$

As before, in case of a tie in probabilities, we assume a nondeterministic choice.

Example 3.4. *Following the computation from Ex. 3.3,* Trace($T$, 18) *terminates in state* ($T\downarrow_{x>10}$, $x \leq 10$, $[x > 10]$). *Point 18 is associated with the trace* $[x > 10]$ *and is classified as black because* cprob($T\downarrow_{x>10}$) = $\langle 0, 1 \rangle$.

## 4. ABSTRACT POISONED SEMANTICS

In this section, we begin by defining a data-poisoning model in which an attacker contributes a number of malicious training items. Then, we demonstrate how to apply the trace-based learner Trace to *abstract sets of training sets*, allowing us to efficiently prove poisoning robustness.

### 4.1. The *n*-poisoning model

We consider a poisoning model where the attacker has contributed up to $n$ elements of the training set—we call it *n-poisoning*. Formally, given a training set $T$ and a natural number $n \leq |T|$, we define the following perturbed set:

$$\Delta_n(T) = \{T' \subseteq T : |T \setminus T'| \leq n\}$$

In other words, $\Delta_n(T)$ captures every training set the attacker could have possibly started from to arrive at $T$.

This definition of dataset poisoning matches many settings studied in the literature.[5,19,23] The idea is that an attacker has contributed a number of malicious data points into the training set to influence the resulting classifier.

We do not know which $n$ points in $T$ are the malicious ones, or if there are malicious points at all. Thus, the set $\Delta_n(T)$ captures every possible subset of $T$ where we have removed up to $n$ (potentially malicious) elements. Our goal is to prove that our classification is robust to up to $n$ possible poisoned points added by the attacker. So if we try every possible dataset in $\Delta_n(T)$ and they all result in the same classification on $x$, then $x$ is robust regardless of the attacker's potential contribution.

Observe that $|\Delta_n(T)| = \sum_{i=1}^{n} \binom{|T|}{i}$. So even for relatively small datasets and number $n$, the set of possibilities is massive, for example, for MNIST-1-7 dataset (§5), for $n = 50$, we have about $10^{141}$ possible training sets in $\Delta_n(T)$.

### 4.2. Abstractions for verifying *n*-poisoning

Our goal is to efficiently evaluate Trace on an input $x$ for all possible training datasets in $\Delta_n(T)$. If all of them yield the same classification $y$, then we know that $x$ is a robust input. Our insight is that we can abstractly interpret Trace on a symbolic set of training sets without having to fully expand it into all of its possible concrete instantiations.

Recall that the state of Trace is $(T, \varphi, \sigma)$; for our purposes, we do not have to consider the sequence of predicates $\sigma$, as we are only interested in the final classification, which is a function of $T$. In this section, we present the *abstract domains* for each component of the learner's state.

**Abstract training sets.** Abstracting training sets is the main novelty of our technique. We use the *abstract element* $\langle T', n' \rangle$ to denote a set of training sets and it captures the definition of $\Delta_{n'}(T')$: For every training set $T'$ and number $n'$, the concretization function—that is, the function that maps an abstract element to the set of concrete elements it represents—is $\gamma(\langle T', n' \rangle) = \Delta_{n'}(T')$. Therefore, we have that initially the *abstraction* function $\alpha$—that is., the function that maps a set of concrete elements to an abstract one—which is defined as $\alpha(\Delta_n(T)) = \langle T, n \rangle$ is precise. Note that an abstract element $\langle T', n' \rangle$ succinctly captures a large number of concrete sets, $\Delta_{n'}(T')$. Further, all operations we perform on $\langle T', n' \rangle$ will only modify $T'$ and $n'$, without resorting to concretization.

Elements in the domain are ordered so that $\langle T_1, n_1 \rangle \sqsubseteq \langle T_2, n_2 \rangle$ if and only if $T_1 \subseteq T_2 \wedge n_1 \leq n_2 - |T_2 \setminus T_1|$.

We can define an *efficient* join operation—that is, the operation that given two abstract elements, returns an abstract element that subsumes both—on two elements in the abstract domain as follows:

Definition 4.1 (Joins). *Given two training sets $T_1$, $T_2$ and $n_1$, $n_2 \in \mathbb{N}$, $\langle T_1, n_1 \rangle \sqcup \langle T_2, n_2 \rangle := \langle T', n' \rangle$ where $T' = T_1 \cup T_2$ and $n' = \max(|T_1 \setminus T_2| + n_2, |T_2 \setminus T_1| + n_1)$.*

Notice that the join of two sets is an overapproximation of the union of the two sets.

Proposition 4.1. *For any $T_1$, $T_2$, $n_1$, $n_2$:*

$$\gamma(\langle T_1, n_1 \rangle) \cup \gamma(\langle T_2, n_2 \rangle) \subseteq \gamma(\langle T_1, n_1 \rangle \sqcup \langle T_2, n_2 \rangle).$$

Example 4.1. *For any training set $T_1$, if we consider the abstract sets $\langle T_1, 2 \rangle$ and $\langle T_1, 3 \rangle$, because the second set represents strictly more concrete training sets, we have*

$$\langle T_1, 2 \rangle \sqcup \langle T_1, 3 \rangle = \langle T_1, 3 \rangle$$

*Now consider the training set $T_2 = \{x_1, x_2\}$. We have*

$$\langle T_2, 2 \rangle \sqcup \langle T_2 \cup \{x_3\}, 2 \rangle = \langle T_2 \cup \{x_3\}, 3 \rangle$$

*Notice how the join increased the poisoned elements from 2 to 3 to accommodate for the additional element $x_3$.*

**Abstract predicates and numeric values.** When abstractly interpreting what predicates the learner might choose for different training sets, we will need to abstractly represent sets of possible predicates. Simply, a set of predicates is abstracted *precisely* as the corresponding set of predicates $\Psi$—that is, for every set $\Psi$, we have $\alpha(\Psi) = \Psi$ and $\gamma(\Psi) = \Psi$. Moreover, $\Psi_1 \sqcup \Psi_2 = \Psi_1 \cup \Psi_2$. For certain operations, it will be handy for $\Psi$ to contain a special null predicate $\Diamond$.

When abstractly interpreting numerical operations, such as cprob and ent, we will need to abstract sets of numerical values. We do so using the standard *intervals* abstract domain (denoted $[l, u]$). For instance, $\alpha(\{0.2, 0.4, 0.6\}) = [0.2, 0.6]$ and $\gamma([0.2, 0.6]) = \{x \mid 0.2 \leq x \leq 0.6\}$. The join of two intervals is defined as $[l_1, u_1] \sqcup [l_2, u_2] = [\min(l_1, l_2), \max(r_1, r_2)]$. Interval arithmetic follows the standard definitions.

### 4.3. Abstract learner Trace#

We are now ready to define an abstract interpretation of the semantics of our decision tree learner, denoted Trace#.

**Abstract domain.** Recall that the state of Trace is $(T, \varphi, \sigma)$; for our purposes, we do not have to consider the sequence of predicates $\sigma$, as we are only interested in the final classification, which is a function of $T$. Using the domains described in Section 4.2, at each point in the learner, our abstract state is a pair $(\langle T', n' \rangle, \Psi')$ that tracks the current set of training sets and the current set of possible most recent predicates the algorithm has split on (for all considered training sets).

When verifying $n$-poisoning for a training set $T$, the initial abstract state of the learner will be the pair $(\langle T, n \rangle, \{\Diamond\})$. In the rest of the section, we define the abstract semantics (i.e., our abstract transformers) for all the operations performed by Trace#. For operations that only affect one element of the state, we assume the other component is unchanged.

### 4.4. Abstract semantics of auxiliary operators

We will begin by defining the abstract semantics of the auxiliary operations in the algorithm before proceeding to the core operations, filter and bestSplit.

We begin with $\langle T, n \rangle \downarrow_{\varphi}^{\#}$, that is, the abstract analog of $T \downarrow_{\varphi}$.

$$\langle T, n\rangle\downarrow^{\#}_{\varphi} := \langle T\downarrow_{\varphi}, \min(n, |T\downarrow_{\varphi}|)\rangle \quad (1)$$

It removes elements not satisfying $\varphi$ from $T$; since the size of $T\downarrow_{\varphi}$ can go below $n$, we take the minimum of the two.

PROPOSITION 4.2. *Let $T' \in \gamma(\langle T, n\rangle)$. For any predicate $\varphi$, we have $T'\downarrow_{\varphi} \in \gamma(\langle T, n\rangle\downarrow^{\#}_{\varphi})$.*

Now, consider cprob($T$), which returns a vector of probabilities for different classes. Its abstract version returns an interval for each probability, denoting the lower and upper bounds based on the training sets in the abstract set:

$$\mathsf{cprob}^{\#}(\langle T, n\rangle) := \left\langle \frac{\left[\max(0, c_i - n), c_i\right]}{\left[|T| - n, |T|\right]} \right\rangle_{i\in[1,k]}$$

where $c_i = |\{(x, i) \in T\}|$. In other words, for each class $i$, we need to consider the best- and worst-case probability based on removing $n$ elements from the training set, as denoted by the denominator and the numerator. Note that in the corner case where $n = |T|$, we set $\mathsf{cprob}^{\#}(\langle T, n\rangle) = \langle[0, 1]\rangle_{i\in[1,k]}$.

PROPOSITION 4.3. *Let $T' \in \gamma(\langle T, n\rangle)$. Then,*

$$\mathsf{cprob}(T') \in \gamma(\mathsf{cprob}^{\#}(\langle T, n\rangle))$$

*where $\gamma(\mathsf{cprob}^{\#}(\langle T, n\rangle))$ is the set of all possible probability vectors in the vector of intervals.*

EXAMPLE 4.2. *Consider the training set on the left side of the tree in Figure 2; call it $T_\ell$. It has 7 white elements and 2 black elements.* $\mathsf{cprob}(T_\ell) = \langle 7/9, 2/9\rangle$, *where the first element is the white probability.* $\mathsf{cprob}^{\#}(\langle T_\ell, 2\rangle)$ *produces the vector* $\langle[5/9, 1], [0, 2/7]\rangle$. *Notice the loss of precision in the lower bound of the first element. If we remove two white elements, we should get a probability of* 5/7, *but the interval domain cannot capture the relation between the numerator and denominator in the definition of* cprob$^{\#}$.

The abstract version of the Gini impurity is identical to the concrete one, except that it performs interval arithmetic:

$$\mathsf{ent}^{\#}(T) = \sum_{i=1}^{k} \iota_i([1, 1] - \iota_i), \quad \text{where } \mathsf{cprob}^{\#}(T) = \langle \iota_1, \ldots, \iota_k\rangle$$

Each term $\iota_i$ denotes an interval.

### 4.5. Abstract semantics of filter

We are now ready to define the abstract version of filter. Since we are dealing with abstract training sets, as well as a set of predicates $\Psi$, we need to consider for each $\varphi \in \Psi$ all cases where $x \models \varphi$ or $x \models \neg\varphi$, and take the join of all the resulting training sets (Definition 4.1). Let

$$\Psi_x = \{\varphi \in \Psi \mid x \models \varphi\} \quad \text{and} \quad \Psi_{\neg x} = \{\varphi \in \Psi \mid x \models \neg\varphi\}$$

Then,

$$\mathsf{filter}^{\#}(\langle T, n\rangle, \Psi, x) := \left(\bigsqcup_{\varphi\in\Psi_x} \langle T, n\rangle\downarrow^{\#}_{\varphi}\right) \sqcup \left(\bigsqcup_{\varphi\in\Psi_{\neg x}} \langle T, n\rangle\downarrow^{\#}_{\neg\varphi}\right)$$

PROPOSITION 4.4. *Let $T' \in \gamma(\langle T, n\rangle)$ and $\varphi' \in \Psi$. Then,*

$$\mathsf{filter}(T', \varphi', x) \in \gamma(\mathsf{filter}^{\#}(\langle T, n\rangle, \Psi, x))$$

EXAMPLE 4.3. *Consider the full dataset $T_{\text{bw}}$ from Figure 2. For readability, we write $T$ instead of $T_{\text{bw}}$ in the example. Let $x$ denote the input with numerical feature 4, and let $\Psi = \{x \le 10\}$. First, note that because $\Psi_{\neg x}$ is the empty set, the right-hand side of the result of applying the* filter$^{\#}$ *operator will be the bottom element $\langle\emptyset, 0\rangle$ (i.e., the identity element for $\sqcup$). Then,*

$$\begin{aligned}\mathsf{filter}^{\#}(\langle T, 2\rangle, \Psi, x) &= \langle T, 2\rangle\downarrow^{\#}_{x\le 10} \sqcup \langle\emptyset, 0\rangle && (\textit{def. of } \mathsf{filter}^{\#})\\ &= \langle T\downarrow_{x\le 10}, 2\rangle \sqcup \langle\emptyset, 0\rangle && (\textit{def. of } \langle T, n\rangle\downarrow^{\#}_{\varphi})\\ &= \langle T\downarrow_{x\le 10}, 2\rangle && (\textit{def. of } \sqcup).\end{aligned}$$

### 4.6. Abstract semantics of bestSplit

We are now ready to define the abstract version of bestSplit. We begin by defining bestSplit$^{\#}$ without handling trivial predicates, then we refine our definition.

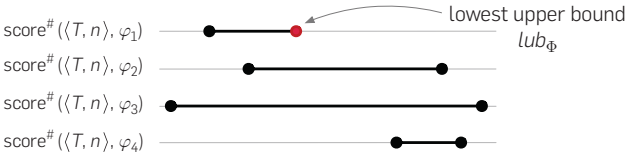
**Minimal intervals.** Recall that in the concrete case, bestSplit returns a predicate that minimizes the function score($T$, $\varphi$). To lift bestSplit to the abstract semantics, we define score$^{\#}$, which returns an interval, and what it means to be a *minimal* interval—that is, the interval corresponding to the abstract minimal value of the objective function score$^{\#}(T, \varphi)$.

Lifting score($T$, $\varphi$) to score$^{\#}(\langle T, n\rangle, \varphi)$ can be done using the sound transformers for the intermediary computations:

$$\begin{aligned}\mathsf{score}^{\#}(\langle T, n\rangle, \varphi) := {} & |\langle T, n\rangle\downarrow^{\#}_{\varphi}| \cdot \mathsf{ent}^{\#}(\langle T, n\rangle\downarrow^{\#}_{\varphi})\\ & + |\langle T, n\rangle\downarrow^{\#}_{\neg\varphi}| \cdot \mathsf{ent}^{\#}(\langle T, n\rangle\downarrow^{\#}_{\neg\varphi})\end{aligned}$$

where $|\langle T, n\rangle| := [|T| - n, |T|]$.

However, given a set of predicates $\Phi$, bestSplit$^{\#}$ must




return the ones with the minimal scores. Before providing the formal definition, we illustrate the idea with an example.

EXAMPLE 4.4. *Imagine a set of predicates $\Phi = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4\}$ with the following intervals for* score$^{\#}(\langle T, n\rangle, \varphi_i)$.
*Notice that $\varphi_1$ has the lowest upper bound for score (denoted in red and named $lub_\Phi$). Therefore, we call* score$^{\#}(\langle T, n\rangle, \varphi_1)$ *the* minimal interval *with respect to* $\Phi$. bestSplit$^{\#}$ *returns all the predicates whose scores* overlap *with the minimal interval* score$^{\#}(\langle T, n\rangle, \varphi_1)$, *which in this case are $\varphi_1$, $\varphi_2$, and $\varphi_3$. This is because there is a chance that $\varphi_1$, $\varphi_2$ and $\varphi_3$ are indeed the predicates with the best score, but our abstraction has lost too much precision for us to tell conclusively.*

Let lb/ub be functions that return the lower/upper bound of an interval. First, we define the lowest upper bound among the abstract scores of the predicates in $\Phi$ as

$$lub_\Phi = \min_{\varphi\in\Phi} \mathrm{ub}(\mathsf{score}^{\#}(\langle T, n\rangle, \varphi))$$

We can now define the set of predicates whose score overlaps with the minimal interval as (we avoid some corner cases here for clarity):

$$\mathsf{bestSplit}^{\#}(\langle T, n\rangle) := \{\varphi \in \Phi \mid \mathsf{lb}\left(\mathsf{score}^{\#}(\langle T, n\rangle, \varphi)\right) \leq lub_{\Phi}\}$$

Lemma 4.1. *Let* $T' \in \gamma(\langle T, n\rangle)$. *Then,*

$$\mathsf{bestSplit}(T') \in \gamma\left(\mathsf{bestSplit}^{\#}(\langle T, n\rangle)\right)$$

### 4.7. Abstracting conditionals

We abstractly interpret conditionals in Trace, as is standard, by taking the join of all abstract states from the feasible *then* and *else* paths. In Trace, there are two branching statements of interest for our purposes, one with the condition $\mathsf{ent}(T) = 0$ and one with $\varphi = \Diamond$.

Let us consider the condition $\varphi = \Diamond$. Given an abstract state $(\langle T, n\rangle, \Psi)$, we simply set $\Psi = \{\Diamond\}$ and propagate the state to the *then* branch (unless, of course, $\Diamond \notin \Psi$, in which case we omit this branch). For $\varphi \neq \Diamond$, we remove $\Diamond$ from $\Psi$ and propagate the resulting state through the *else* branch.

Next, consider the conditional $\mathsf{ent}(T) = 0$. For the *then* branch, we need to *restrict* an abstract state $(\langle T, n\rangle, \Psi)$ to training sets with 0 entropy: Intuitively, this occurs when all elements have the same classification. We ask: *Are there any concretizations composed of elements of the same class?*, and we proceed through the *then* branch with the following training set abstraction:

$$\bigsqcup_{i \in [1,k]} pure(\langle T, n\rangle, i)$$

$$pure(\langle T, n\rangle, i) := \text{Let } T' = \{(x, y) \in T \mid y = i\} \text{ in } \\ \text{if } |T \setminus T'| \leq n \text{ then } \langle T', n - |T \setminus T'|\rangle \\ \text{else } \bot$$

The idea is as follows: The set $T'$ defines a subset of $T$ containing only elements of class $i$. But if we have to remove more than $n$ elements from $T$ to arrive at $T'$, then the conditional is not realizable by a concrete training set of class $i$, and so we return the empty abstract state.

In the case of $\mathsf{ent}(T) \neq 0$ (*else* branch), we soundly (imprecisely) propagate the original state without restriction.

### 4.8. Soundness of abstract learner

Trace# soundly overapproximates the results of Trace and can therefore be used to prove robustness to $n$-poisoning. Let $I = ([l_1, u_2], \ldots, [l_k, u_k])$ be a set of intervals. We say that interval $[l_i, u_i]$ dominates $I$ if and only if $l_i > u_j$ for every $j \neq i$.

Theorem 4.2. *Let* $\langle T', n'\rangle$ *be the final abstract state of* $\mathsf{Trace}^{\#}(\langle T, n\rangle, x)$. *If* $I = \mathsf{cprob}^{\#}(\langle T', n'\rangle)$) *and there exists an interval in I that dominates I (i.e., same class is selected for every* $T \in \gamma(\langle T, n\rangle)$*), then x is robust to n-poisoning of T.*

### 4.9. Disjunctive abstraction

The state abstraction used by Trace# can be *imprecise*, mainly due to the join operations that take place, for example, during filter#. The primary concern is that we are forced to perform a very imprecise join between possibly quite dissimilar training set fragments. Consider the following example:

Example 4.5. *Let us return to* $T_{\mathrm{bw}}$ *from Figure 2, but imagine we have continued the computation after filtering using* $x \leq 10$ *and have selected some best predicates. Specifically, consider a case in which we have* $x = 4$ *and*

- $\langle T, 1\rangle$, *where* $T = \{0, 1, 2, 3, 4, 7, 8, 9, 10\}$
- $\Psi = \{x \leq 3, x \leq 4\}$ *(ignoring whether this is correct)*

*Let us evaluate* $\mathsf{filter}^{\#}(\langle T, 1\rangle, \Psi, x)$. *Following the definition of* filter#, *we will compute*

$$\langle T', n'\rangle = \langle T_{\leq 4}, 1\rangle \sqcup \langle T_{>3}, 1\rangle$$

*where* $T_{\leq 4} = \{(4, b), (3, w), (2, w), (1, w), (0, b)\}$ *and* $T_{>3} = \{(4, b), (7, w), (8, w), (9, w), (10, w)\}$, *thus giving us* $T' = T$ *(the set we began with) and* $n' = 5$ *(much larger than what we began with). This is a large loss in precision.*

To address this imprecision, we will consider a *disjunctive* version of our abstract domain, consisting of unboundedly many disjuncts of this previous domain, which we represent as a set $\{(\langle T, n\rangle_i, \Psi_i)\}_i$. Our join operation becomes very simple: It is the union of the two sets of disjuncts.

Definition 4.2 (Joins). *Given two disjunctive abstractions* $D_I = \{(\langle T, n\rangle_i, \Psi_i)\}_{i \in I}$ *and* $D_J = \{(\langle T, n\rangle_j, \Psi_j)\}_{j \in J}$, *we define*

$$D_I \sqcup D_J := D_I \cup D_J$$

Adapting Trace# to operate on this domain is immediate: Each of the transformers described in the previous section is applied to each disjunct. Because our disjunctive domain eschews memory efficiency and time efficiency for precision, we are able to prove more things, but at a cost.

## 5. IMPLEMENTATION AND EVALUATION

We implemented our algorithms Trace and Trace# in C++ in a (single-threaded) prototype we call Antidote. Our evaluation aims to answer the following research questions:

**RQ1** Can Antidote prove data-poisoning robustness for real-world datasets?

**RQ2** How does the performance of Antidote vary with respect to the scale of the problem and the choice of abstract domain?

### 5.1. Benchmarks and experimental setup

We experiment on 5 datasets (Table 1). We obtained the first three datasets from the UCI Machine Learning Repository.[7] Iris is a small dataset that categorizes three related flower species; Mammographic Masses and Wisconsin Diagnostic Breast Cancer are the two datasets of differing complexities related to classifying whether tumors are cancerous. We also evaluate the widely studied MNIST dataset of handwritten digits,[11] which consists of 70,000 grayscale images (60,000 training, 10,000 test) of the digits zero through nine. We

consider a form of MNIST that has been used in the poisoning literature and create another variant for evaluation:

- We make the same simplification as in other work on data poisoning[2,19] and restrict ourselves to the classification of ones versus sevens (13,007 training instances and 2163 test instances), which we denote MNIST-1-7-Real. This benchmark has been recently used to study poisoning in support vector machines.[19]
- Each MNIST-1-7-Real image's pixels are 8-bit integers (which we treat as real-valued); to create a variant of the problem with reduced scale, we *also* consider MNIST-1-7-Binary, a black-and-white version that uses each pixel's most significant bit (i.e., our predicates are Boolean).

For each dataset, we consider a decision tree learner with a maximum tree depth (i.e. the number of calls to bestSplit) ranging from 1 to 4. Table 1 shows that test set accuracies of the decision trees learned by Trace are reasonably high—affirmation that when we prove the robustness of its results, we are proving something worthwhile.

**Experimental setup.** For each test element, we explore the amount of poisoning (i.e., how large of a $n$ from our $\Delta_n$ model) for which we can prove the robustness property as follows.

1. For each combination of dataset $T$ and tree depth $d$, we begin with a poisoning amount $n = 1$; that is, a single element could be missing from the training set.
2. For each test set element $x$, we attempt to prove that $x$ is robust to poisoning $T$ using any set in $\Delta_n(T)$. Let $S_n$ be the test subset for which we do prove robustness for poisoning amount $n$. If $S_n$ is non-empty, we double $n$ and again attempt to verify the property on elements in $S_n$.
3. If at a depth $n$ all instances fail, we binary search between $n$ and $n/2$ to find an $n/2 < n' < n$ at which some instances terminate. This approach allows us to better illustrate the experiment trends in our plots.

Failure occurs due to any of three cases: (*i*) The computed over-approximation does not conclusively prove robustness, (*ii*) the computation runs out of memory, or (*iii*) the computation exceeds a one-hour time-out. We run the entire procedure for the non-disjunctive and disjunctive abstract domains.

### 5.2. Effectiveness of antidote

We evaluate how effective Antidote is at proving data-poisoning robustness. In this experiment, we consider a run of the algorithm on a single test element successful if either the non-disjunctive or disjunctive abstract domain succeeds at proving robustness (mimicking a setting in which two instances of Trace$^{\#}$, one for each abstract domain, are run in parallel)—we contrast the results for the different domains in the original paper. Figure 6 shows these results.

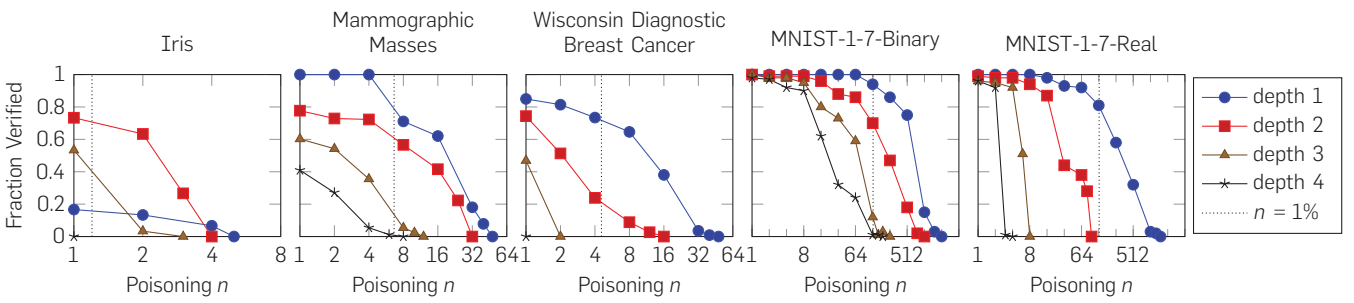
To exemplify the power of Antidote, draw your attention to the depth-2 instance of Trace$^{\#}$ invoked on MNIST-1-7-Real. For 38/100 test instances, we are able to verify that even if the training set had been poisoned by an attacker who contributed up to 64 poisoned elements $\approx \frac{1}{2}\%$, the attacker would not have had any power to change the resulting classification. Conventional machine learning wisdom says that, in decision tree learning, small changes to the training set can cause the model to behave quite differently. Our results

**Table 1. Detailed metrics for the benchmark datasets considered in our evaluation.**

| | Size | | | | DT Test-Set Accuracy (%) | | | |
|---|---|---|---|---|---|---|---|---|
| **Dataset** | **Training** | **Test** | **Features $\mathcal{X}$** | **Classes $\mathcal{Y}$** | **Depth 1** | **2** | **3** | **4** |
| Iris | 120 | 30 | $\mathbb{R}^4$ | {Setosa, Versicolour, Virginica} | 20.0 | 90.0 | 90.0 | 90.0 |
| Mammographic masses | 664 | 166 | $\mathbb{R}^5$ | {benign, malignant} | 80.7 | 83.1 | 81.9 | 80.7 |
| Wisconsin diagnostic Breast cancer | 456 | 113 | $\mathbb{R}^{30}$ | {benign, malignant} | 91.2 | 92.0 | 92.9 | 94.7 |
| MNIST-1-7-Binary | 13,007 | 100* | $\{0,1\}^{784}$ | {one, seven} | 95.7 | 97.4 | 97.8 | 98.3 |
| MNIST-1-7-Real | 13,007 | 100* | $\mathbb{R}^{784}$ | {one, seven} | 95.6 | 97.6 | 98.3 | 98.7 |

* Test set accuracy for MNIST is computed on the full 2163 instances; robustness experiments are performed on 100 randomly chosen test set elements.

**Figure 6. Fraction of test instances proven robust versus poisoning parameter *n* (log scale). The dotted line is a visual aid, indicating *n* is 1% of the training set size.**



verify nuance—sometimes, there is some stability. These 38 verified instances average ∼800*s* run time. $\Delta_{64}(T)$ consists of $\sim 10^{174}$ concrete training sets; this is staggeringly efficient compared to a naïve enumeration baseline, which would be unable to verify robustness at this scale.

To answer **RQ1**, *Antidote can verify robustness for real-world datasets with extremely large perturbed sets and decision tree learners with high accuracies*.

We delegate the results for **RQ2** to the original paper, but we note that decision tree depth the number of poisoned elements are the most important factors for the scalability of our approach: Larger trees and more poisoning mean a higher loss of precision, and therefore less proofs. Similarly, larger trees lead to scalability issues as the number of abstract datasets we maintain may grow exponentially.

## 6. RELATED WORK

**Data poisoning.** Data-poisoning robustness has been studied extensively from an attacker perspective.[2, 13, 15, 24, 25] This body of work has demonstrated attacks that can degrade classifier (in particular, SVMs) accuracy, sometimes dramatically. Our approach instead *proves* that no poisoning attack exists using abstract interpretation, while existing techniques largely provide search techniques for finding poisoned training sets. Recently, we have showed that our technique is robust and can handle different poisoning models,[14] for example, ones in which the attacker can remove, add, or modify data. This recent work also shows how poisoning can disparately impact different parts of the test data (e.g., when dividing a test data where each element corresponds to a person based on ethnicity), therefore opening new problems in the field of algorithmic fairness.

Some recent techniques modify the training processes and hypothesis class to prove robustness to data poisoning,[12,18] while our approach proves that a certain algorithm is robust to poisoning with respect to a given data set. Additionally, these approaches provide *probabilistic* guarantees about certain kinds of attacks on a modified training algorithm, while our work provides deterministic guarantees.

**Abstract interpretation for robustness.** Abstract interpretation[6] is a framework for static program analysis. Our work is inspired by abstract interpretation-based verification of neural networks.[1] However, we tackle the problem of verifying training time robustness, while existing works focus on test-time robustness. The former problem requires abstracting sets of training sets, while the latter only requires abstracting sets of individual inputs.


## Acknowledgments

This material is based upon work supported by the National Science Foundation under grant numbers 1652140, 1704117, 1750965, and 1918211. The work is also supported by a Facebook Award and a Microsoft Faculty fellowship. C



**References**

1. Albarghouthi, A. Introduction to neural network verification. arXiv:2109.10317, 2021.
2. Biggio, B., Nelson, B., Laskov, P. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Conference on Machine Learning*, ICML'12, Omnipress.
3. Breiman, L. *Classification and Regression Trees*, Routledge, 2017.
4. Carlini, N., Wagner, D. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 39–57.
5. Chen, X., Liu, C., Li, B., Lu, K., Song, D. Targeted backdoor attacks on deep learning systems using data poisoning. arXiv:1712.05526, 2017.
6. Cousot, P., Cousot, R. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints, 1977.
7. Dua, D., Graff, C. UCI machine learning repository, 2017.
8. Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., Vechev, M. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, May 2018, 3–18.
9. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*. Springer, 2017, 97–117.
10. Laishram, R., Phoha, V.V. Curie: A method for protecting SVM classifier from poisoning attack. *CoRR*, abs/1606.01584, 2016.
11. LeCun, Y., Cortes, C., Burges, C.J.C. The MNIST database of handwritten digits.
12. Levine, A., Feizi, S. Deep partition aggregation: Provable defenses against general poisoning attacks. In *International Conference on Learning Representations*, 2020.
13. Mei, S., Zhu, X. Using machine teaching to identify optimal training-set attacks on machine learners. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, AAAI'15. AAAI Press, 2015, 2871–2877.
14. Meyer, A.P., Albarghouthi, A., D'Antoni, L. Certifying robustness to programmable data bias in decision trees. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems*, December 6-14, 2021.
15. Newell, A., Potharaju, R., Xiang, L., Nita-Rotaru, C. On the practicality of integrity attacks on document-level sentiment analysis. In *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop* (New York, NY, USA). ACM, 83–93.
16. Quinlan, J.R. Induction of decision trees. *Mach. Learn 1*, 1 (1986), 81–106.
17. Quinlan, J.R. C4.5: Programs for machine learning. *The Morgan Kaufmann Series in Machine Learning*, Morgan Kaufmann, San Mateo, CA, 1993.
18. Rosenfeld, E., Winston, E., Ravikumar, P., Kolter, Z. Certified robustness to label-flipping attacks via randomized smoothing. In *International Conference on Machine Learning*. PMLR, 2020, 8230–8241.
19. Steinhardt, J., Koh, P.W.W., Liang, P.S. Certified defenses for data poisoning attacks. In *Advances in Neural Information Processing Systems*, 2017, 3517–3529.
20. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., Fergus, R. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations*, Banff, AB, Canada, April 14–16, 2014, Conference Track Proceedings.
21. Wang, S., Pei, K., Whitehouse, J., Yang, J., Jana, S. Formal security analysis of neural networks using symbolic intervals. In *27th [USENIX] Security Symposium*. 2018, 1599–1614.
22. Wang, Y., Jha, S., Chaudhuri, K. Analyzing the robustness of nearest neighbors to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018* (Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018), 5120–5129.
23. Xiao, H., Biggio, B., Brown, G., Fumera, G., Eckert, C., Roli, F. Is feature selection secure against training data poisoning? In *International Conference on Machine Learning*, 2015, 1689–1698.
24. Xiao, H., Biggio, B., Nelson, B., Xiao, H., Eckert, C., Roli, F. Support vector machines under adversarial label contamination. *Neurocomput. 160*, C (July 2015), 53–62.
25. Xiao, H., Xiao, H., Eckert, C. Adversarial label flips attack on support vector machines. In *Proceedings of the 20th European Conference on Artificial Intelligence*, ECAI'12 (Amsterdam, The Netherlands, 2012), IOS Press, 870–875.



**Samuel Drews, Loris D'Antoni**([sedrews, ldantoni]@wisc.edu), University of Wisconsin-Madison, USA.

**Aws Albarghouthi**(aws@cs.wisc.edu), University of Wisconsin-Madison, USA.

**University of Central Florida**
***Multiple Positions for Assistant, Associate or Full Professor, Artificial Intelligence Initiative***

The Artificial Intelligence Initiative (Aii) at the University of Central Florida (UCF) is accepting applications from strong candidates for multiple 9-month, full-time faculty positions at the rank of assistant professor (tenure-earning) and associate professor or professor (tenured) in core areas of AI and their applications including: Computer Vision, Natural Language Processing, Robotics, Machine Learning, Data Analytics, FinTech, Smart Cities, Connected and Automated Vehicles, Cyber Security, Mathematical Aspects of Deep Learning, Theory of AI and Data Science, Brian-Inspired Ai, Biomedical Applications, Smart Materials, Smart Mobility, Genomics and Computational Biology, as well as Innovative Computing domains including but not limited to Optical Computing, Neuromorphic Computing and AI in Next generation of Wireless Communication Systems. Exceptional candidates only will be considered for appointment at full professor level.

Aii is a multi-college initiative at UCF involving the Colleges of Engineering and Computer Science, Sciences, Medicine, Business, and Optics and Photonics. Candidates with publications in the most selective conferences and journals are strongly encouraged to apply. We anticipate that close to thirty new AI faculty members will be hired, with qualified candidates tenured in corresponding colleges. The position will carry a rank commensurate with the candidate's prior experience and record.

UCF is one of the nation's largest universities with a diverse student body of more than 70,000 students. UCF has grown substantially in size, quality, diversity, and reputation in its first 50 years. Today, the university offers more than 230 degree programs. UCF is an economic engine, attracting and supporting industries vital to the region's future while providing students with real-world experiences that help them succeed after graduation.

**Minimum Qualifications:**
A Ph.D., M.D., or equivalent degree from an accredited institution in an area appropriate to this position at the time of the appointment.

To be eligible for appointment as a tenured associate professor or professor upon hire, the selected candidate must have a demonstrated record of teaching, research, and service commensurate with a tenured faculty appointment at the rank of associate professor or professor, in applicable tenure home.

**Preferred Qualifications:**
- ▸ Highly recognized contributions and leadership in the area(s) of expertise.
- ▸ Demonstrated strong research publication record in the most selective conferences and journals (such as those included in csrankings.org or flagship venues of the respective areas).
- ▸ Effective teaching skills, and ability to effectively communicate with students in both large and small audiences.
- ▸ High potential to initiate and obtain funding for the research programs, appropriate for the field of study.

To apply, refer to https://www.ucf.edu/jobs/ and search for **job announcement R102905**. In addition to the online application, interested candidates should upload a cover letter, a current curriculum vitae, and a list with contact information for three (3) professional references.

For more information, contact Linda Lockey, Administrative Support, at Linda.Lockey@ucf.edu.

**Equal Employment Opportunity Statement**

The University of Central Florida is an equal opportunity/affirmative action employer. All qualified applicants will receive consideration for employment without regard to sex, gender identity, sexual orientation, race, color, religion, national origin, disability, protected Veteran status, age, or any other characteristic protected by law. UCF's Equal Opportunity Statement can be viewed at: http://www.oie.ucf.edu/documents/PresidentsStatement.pdf. As a Florida public university, UCF makes all application materials and selection procedures available to the public upon request. The UCF's affirmative action plans for qualified individuals with disabilities and protected Veterans are available for inspection in the Office for Institutional Equity, Monday through Friday, from 9:00 a.m. to 5:00 p.m., upon request.

[CONTINUED FROM P. 116] which were controlled by a musician operating a keyboard of large levers.

While ordinary people listened to the music of the hydraulis, Sulla imagined how the mechanism might be used to control the prices merchants charged for their goods. Every week, the government could state the price for many standard items, and merchants throughout the Roman Republic would charge exactly those amounts. How the hydraulis would contribute to that process was not yet clear, but Sulla suggested that a few hundred of these devices in the Laboratorium could calculate appropriate prices within the current government policy, while considering natural forces such as the seasons of the year.

Sulla placed Andivius in temporary command of his great yacht, named Publius after his grandfather, who ironically was an advocate for democracy, with Athens as its obvious first destination. Having some experience yachting, Andivius was able to direct its proletarian sailors, although he detected they sometimes followed their own instincts rather than his orders. The only literate member of the crew, Proditor, seemed interested in Sulla's project, so Andivius took him into Athens as his servant.

Their initial goal was collecting various *clepsydra* water clocks, which had different designs, especially in the feedback mechanism that regulated the flow of water. Sundials were a marvelous ancient invention, which both the Egyptians and Babylonians laid claim to. Their obvious defect is that they do not work at night, so some still-unidentified civilization invented clocks based on the slow dripping of water from one container to another. Also, except at the literal equinox, hours in the day as measured by sundials were longer in summer than in winter, given that there were always 12 hours from dawn to dusk. The clever Greeks noticed that time, as measured by water clocks, was not much more stable than when dictated by the sun. So, they invented a feedback device that controlled constant water drippage. Ctesibius, a Greek living in Egypt, perfected the clepsydra and invented the hydraulis.

While they were gathering clepsydras, Proditor mentioned that one of the collections they had visited included a very different device, apparently based on ideas about the use of compressed air, which had been developed by Ctesibius. Called an *aeolipile*, it was an engine that used steam to spin a part of the device around, visually dramatic but having no practical purpose. Proditor asked Andivius whether some of the devices they gathered might be adapted for social progress rather than governmental control. Andivius joked, "Oh, are you suggesting that we use hot steam rather than cool sails and enslaved oarsmen to move our Publius boat even faster?" Andivius laughed first, followed by Proditor.

**He gave them the most marvelous computer of the ancient world, a complex system of gears that could predict the motions of the moon and planets.**

After completing their collection of relics and documents in Athens, they headed for the island of Rhodes, which had recently become more culturally significant even than Athens. They visited Posidonius, who operated the leading school in the western world, had already done an exploratory tour around the Mediterranean, and had written a scientific analysis of how the movement of the Moon produced the complex tides in the sea. Under some political pressure and offered a decent price, he gave them the most marvelous computer of the ancient world, a complex system of gears that could predict the motions of the moon and planets. Greek inscriptions coded the signs of the zodiac and the months of the Egyptian year.

As Publius sailed away from Rhodes, south toward Alexandria in Egypt, where they hoped to find more relics left by the long-deceased Ctesibius, Andivius lectured Proditor about how the Roman government might use this astronomical device. For example, it could control the work by farmers, over the course of each year, following rules for their location and crops. Proditor seemed a bit distracted, and commented, "Yes, that is a crucial insight, sir. But I was thinking about what Posidonius said about the excellent *computatra* being developed on the nearby island of Antikythera. Shouldn't we stop there before going to antique Alexandria?" Andivius did not recall Posidonius mentioning Antikythera, but became convinced by Proditor's enthusiasm, so he ordered the crew to head westward rather than south.

The truth suddenly became clear approximately a mile from the island. Without any command from Andivius, the crew dropped the sails and prepared the two small dinghies used to reach land where no port existed. Proditor and two other members of the crew approached Andivius with daggers in their hands.

"Sorry about this," Proditor said, "but Antikythera has no computatra but instead is held by pirates. I'm not sorry that we must now sink this ship, so Sulla never learns what happened and comes after us in revenge."

Three stabs and Andivius was dead. Proditor and the other crewmen took whatever valuables they could loot but not any of the relics, which they considered valueless, and set fire to Publius. We do not know their fate when they attempted to join the Antikythera pirates, but the amazing astronomical device did survive. Nearly two thousand years later, in 1901, the real but incredible ancient computer now called the Antikythera Mechanism was accidentally discovered by divers in the shipwreck. Whether it could have saved Rome from its decline and fall remains a mystery.

**William Sims Bainbridge** (wsbainbridge@hotmail.com) is a sociologist who taught classes on crime and deviant behavior at respectable universities before morphing into a computer scientist, editing an encyclopedia of human-computer interaction, writing many books on things computational, from neural nets to virtual worlds to personality capture, then repenting and writing harmless fiction.

From the intersection of computational science and technological speculation, with boundaries limited only by our ability to imagine what could be.

DOI:10.1145/3578622

William Sims Bainbridge

# Future Tense
# Aftermath Impact

*An ancient Roman dispatched to find the greatest technological advances of the time may lose something of far greater importance.*

WITH GREAT FOREBODING, Andivius rushed toward the Amphitheatrum, terrified that dictator Lucius Cornelius Sulla might have ordered him there for purpose of *supplicium*, which could mean anything from supplication to execution. Rome had entered a period of deep uncertainty, decades before Caesar and Augustus established the Empire, and democracy was disintegrating into chaos that ironically the unstable Sulla had promised to reorganize.

"Ah, Andivius, *ave atque vale*!"

Horrors! Sulla had just spoken the equivalent of "*hail and farewell*," which he might well say to a servant who was about to be murdered for his poor quality of work.

"I am sending you on a glorious expedition, but first I must instruct you on the meaning of its goal. We all know that democracy was fatally erratic, so we need a new machinery of government, literally *intellegentias artificialis*, a set of machines that make decisions on the basis of strict rules."

Sulla then held up his pocket abacus, a Roman improvement over the Greek version of this already ancient calculating device, now used regularly by merchants and tax collectors. He explained a plan to create beside the Senate a new Laboratorium that would combine all the science and invention of Greece, Egypt, and the rest, to give Rome a vastly superior culture. So, Sulla ordered Andivius to sail the sea between Greece and Egypt, collecting artifacts that represented potentially great computational inventions, thereby assembling the equipment for aggressive analysis and innovation in the Laboratorium.

IMAGE BY JUSTIN HENDERSON

**"Democracy was fatally erratic, so we need a new machinery of government, literally *intellegentias artificialis*."**

Before dismissing Andivius, Sulla showed him why they were meeting at the Amphitheatrum, an open arena for gladiators and festivals. Music to excite the audience was provided by a complicated machine called a *hydraulis*, the ancient ancestor of Renaissance church organs. It was a very appropriate metaphor for the future of government, because slaves labored to pump air into a huge reservoir that was placed in a cistern of water. That had the effect of smoothing out the surges from slavish pumping and providing constant-pressure air for the organ's pipes, [CONTINUED ON P. 115]


*March 13–17 · Tokyo · Japan*

**The Art, Science, and Engineering of Programming**

**General Chair**
Shigeru Chiba *U Tokyo*
**Program Chair**
Robert Hirschfeld *HPI U Potsdam*
**Organizing Chair**
Tomoharu Ugawa *U Tokyo*
**Artifact Evaluation Chairs**
Patrick Rein *HPI U Potsdam*
Fabio Niephaus *Oracle Labs*
**Workshops Chairs**
Elisa Gonzalez Boix *VU Brussel*
Youyou Cong *Tokyo Tech*

acm In-Cooperation
acm SIGPLAN
SIGSOFT
AOSA

https://2023.programming-conference.org