COMMUNICATIONS OF THE ACM

CACM.ACM.ORG

06/2019 VOL.62 NO.06

Geoffrey Hinton, Yoshua Bengio, and Yann LeCun

Recipients of ACM's A.M. Turing Award

Association for Computing Machinery

DREAM ZONE!

SIGGRAPH ASIA 2019 BRISBANE

The 12th ACM SIGGRAPH Conference and Exhibition on Computer Graphics and Interactive Techniques in Asia

Conference 17 - 20 November 2019
Exhibition 18 - 20 November 2019

Brisbane Convention & Exhibition Centre (BCEC),
Brisbane, Australia

Sponsored by:

acm

Organized by

koelnmesse
we energize your business | since 1924

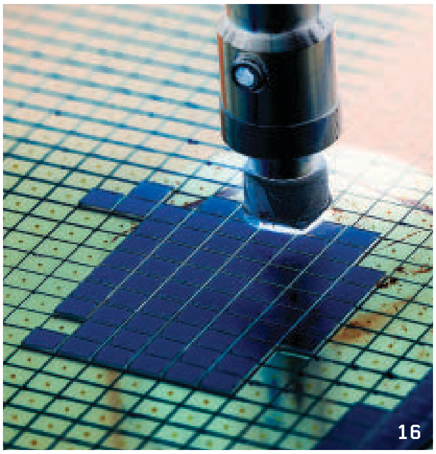
COMMUNICATIONS OF THE ACM

Departments

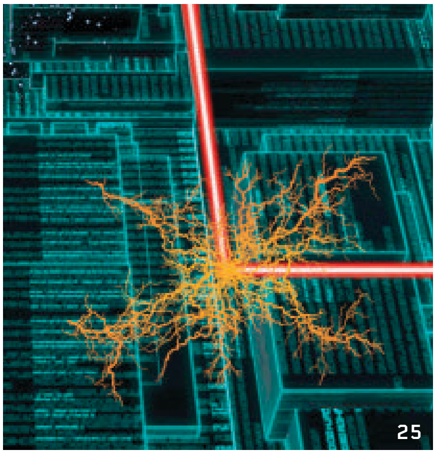
Last Byte

Watch the recipients discuss this work in the exclusive *Communications* video. https://cacm.acm.org/videos/2018-acm-turing-award

News



Viewpoints

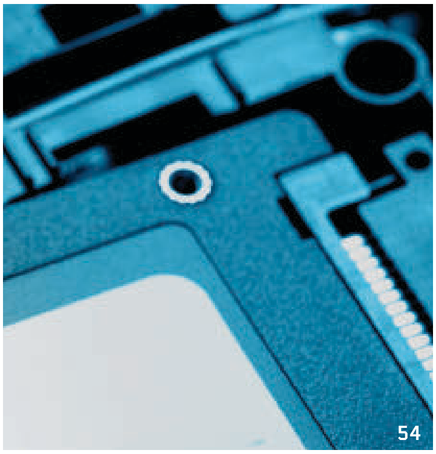


IMAGES BY: (L) MACRO PHOTO; (R) ANDRIJ BORYS ASSOCIATES, USING SHUTTERSTOCK

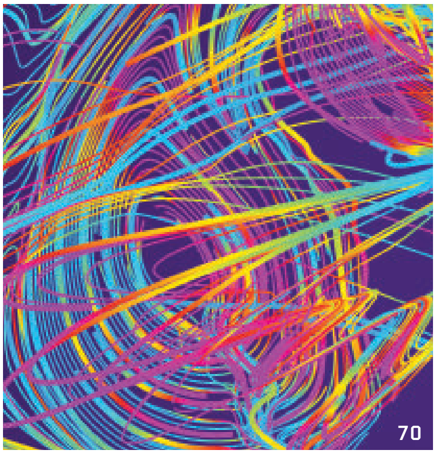
Practice

Articles' development led by acmqueue
queue.acm.org

Contributed Articles

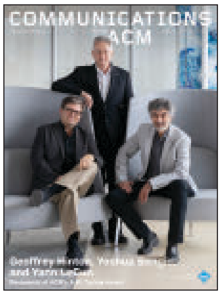


Review Articles



Watch the authors discuss this work in the exclusive *Communications* video. https://cacm.acm.org/videos/the-challenge-of-crafting-intelligible-intelligence

Research Highlights



About the Cover:
The recipients of the 2018 ACM A.M. Turing Award (from left) Yann LeCun, Geoffrey Hinton, and Yoshua Bengio photographed at the Vector Institute in Toronto, Canada, on April 11, 2019. Photographer: Alexander Berg, https://www.alexberg.com/

Association for Computing Machinery
Advancing Computing as a Science & Profession

IMAGES BY: (L) ANTON JANKOVOY; (C, R) FROM SHUTTERSTOCK.COM

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.

ACM U.S. Technology Policy Office
Adam Eisgrau,
Director of Global Policy and Public Affairs
1701 Pennsylvania Ave NW, Suite 200,
Washington, DC 20006 USA
T (202) 580-6555; acmpo@acm.org

Computer Science Teachers Association
Jake Baskin
Executive Director

STAFF

DIRECTOR OF PUBLICATIONS
Scott E. Delman
cacm-publisher@cacm.acm.org

Executive Editor
Diane Crawford
Managing Editor
Thomas E. Lambert
Senior Editor
Andrew Rosenbloom
Senior Editor/News
Lawrence M. Fisher
Web Editor
David Roman
Editorial Assistant
Danbi Yu

Art Director
Andrij Borys
Associate Art Director
Margaret Gray
Assistant Art Director
Mia Angelica Balaquiot
Production Manager
Bernadette Shade
Intellectual Property Rights Coordinator
Barbara Ryan
Advertising Sales Account Manager
Ilia Rodriguez

Columnists
David Anderson; Michael Cusumano; Peter J. Denning; Mark Guzdial; Thomas Haigh; Leah Hoffmann; Mari Sako; Pamela Samuelson; Marshall Van Alstyne

CONTACT POINTS

Copyright permission
permissions@hq.acm.org
Calendar items
calendar@cacm.acm.org
Change of address
acmhelp@acm.org
Letters to the Editor
letters@cacm.acm.org

WEBSITE

http://cacm.acm.org

AUTHOR GUIDELINES

http://cacm.acm.org/about-communications/author-center

ACM ADVERTISING DEPARTMENT

2 Penn Plaza, Suite 701, New York, NY 10121-0701
T (212) 626-0686
F (212) 869-0481

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

Media Kit acmmediasales@acm.org

Association for Computing Machinery (ACM)
2 Penn Plaza, Suite 701
New York, NY 10121-0701 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.

ACM Media Advertising Policy
Communications of the ACM and other ACM Media publications accept advertising in both print and electronic formats. All advertising in ACM Media publications is at the discretion of ACM and is intended to provide financial support for the various activities and services for ACM members. Current advertising rates can be found by visiting http://www.acm-media.org or by contacting ACM Media Sales at (212) 626-0686.

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

COMMUNICATIONS OF THE ACM

(ISSN 0001-0782) is published monthly by ACM Media, 2 Penn Plaza, Suite 701, New York, NY 10121-0701. Periodicals postage paid at New York, NY 10001, and other mailing offices.

POSTMASTER

Please send address changes to
Communications of the ACM
2 Penn Plaza, Suite 701
New York, NY 10121-0701 USA

Printed in the USA.

Association for Computing Machinery

PLEASE RECYCLE THIS MAGAZINE

DOI:10.1145/3326069

Cherri M. Pancake

ACM Awards Honor CS Contributions

In this issue of *Communications*, as evidenced by the cover and lead article, we celebrate the latest recipients of the **ACM A.M. Turing Award**. Yoshua Bengio, Yann LeCun, and Geoffrey Hinton carried out pioneering work in deep learning that has touched all our lives. As Turing Laureates, they now join the eminent group of technology visionaries recognized with the world's highest distinction in computing.

The Turing Award is one of a suite of professional honors ACM bestows annually to recognize technical achievements that have made significant contributions to our field. This month, I will have the pleasure of joining the awardees, ACM Fellows, and other luminaries in San Francisco for the ACM Awards Banquet. The annual event pays tribute to computing excellence and to those whose contributions and innovations have had a lasting impact on our field.

Among the new honorees is Shwetak Patel, winner of the **ACM Prize in Computing**. This award recognizes individuals who have made significant contributions during the early years of their careers. Patel is being honored for his innovative work in applying sensor systems to problems of sustainability and health care. Also on hand will be Mendel Rosenblum, being honored as the first winner of the **ACM Charles P. "Chuck" Thacker Breakthrough in Computing Award**. This new biennial award recognizes individuals whose work exemplifies "out-of-the-box" thinking. Rosenblum's work echoes Thacker's trademark can-do approach: he reinvented the virtual machine concept, thereby revolutionizing datacenters and making today's cloud computing possible.

Pavel Pevzner receives the **ACM Paris Kanellakis Theory and Practice Award**, recognizing theoretical advances that have had a demonstrable effect on computing practice. Pevzner pioneered algorithms for rapidly sequencing DNA; his algorithms underlie almost all sequence assemblers used today and were used to reconstruct the vast majority of genomic sequences available in databases. The **ACM Grace Murray Hopper Award** honors a computing professional who has made a major technical or service contribution by the age of 35. This year, two individuals are being recognized: Michael J. Freedman for the design and deployment of self-organizing peer-to-peer systems; and Constantinos Daskalakis for his contributions to complexity and game theory.

Gerald C. Combs is being recognized with the **ACM Software System Award**, given to an institution or individual(s) for developing a software system of lasting influence. He created the WireShark network protocol analyzer, used by practitioners and researchers worldwide to analyze and troubleshoot a wide range of network protocols. The **2019–2020 ACM Athena Lecturer Award**, a biennial honor celebrating fundamental CS contributions by women researchers, goes to Elisa Bertino in recognition of her groundbreaking work in data security and privacy. Chelsea Finn from UC Berkeley receives the **ACM Doctoral Dissertation Award** for her work on "Learning to Learn with Gradients."

The **ACM Distinguished Service Award**, which celebrates service contributions to the computing community, goes to Paramir (Victor) Bahl, for his work founding conferences, publications, and a SIG for researchers and practitioners in the mobile and wireless networking community, as well as contributions to technology policy. Robert Sedgewick is being honored with the **ACM Karl V. Karlstrom Outstanding Educator Award** for the outstanding textbooks and online materials he created, which are used worldwide for courses in introductory computer science. Chris Stephenson is receiving the **Outstanding Contribution to ACM Award** for her landmark work in bringing K–12 teachers worldwide the tools and resources needed to introduce computer science to future generations. The recipient of the **ACM Eugene L. Lawler Award** for Humanitarian Contributions within Computer Science and Informatics is Meenakshi Balakrishnan for developing cost-effective solutions to address the special mobility and education challenges of the visually impaired in developing countries. The **ACM-AAAI Allen Newell Award**, presented to an individual for career contributions that have breadth within CS or that bridge CS and other disciplines, has been awarded to Henry Kautz for his work at the intersection of AI, computational social science, and public health.

Last but not least, the Awards Banquet will celebrate 56 incoming **ACM Fellows**. A complete list of names and their key achievements can be found at https://awards.acm.org/fellows.

The prestige of ACM's awards brings global attention to outstanding technical and professional achievements throughout the computing community. We *all* benefit when fine work and lasting accomplishments in computer science are celebrated. I hope you will participate this coming year, by making sure the key achievers in your own area are nominated. Our award committees, led by Awards Co-Chairs John White and Vinton Cerf, do an outstanding job, but they rely on people like you to identify and put forward strong candidates. Learn more at https://awards.acm.org/award-nominations.

Cherri M. Pancake is President of ACM, professor emeritus of electrical engineering and computer science, and director of a research center at Oregon State University, Corvallis, OR, USA.

ACM-IMS Data Science Summit

June 15, 2019 | Palace Hotel, San Francisco

An interdisciplinary event bringing together researchers and practitioners to address deep learning, reinforcement learning, robustness, fairness, ethics, and the future of data science.

Computing and statistics underpin the rapid emergence of data science as a pivotal academic discipline. ACM and IMS—the Institute of Mathematical Statistics—the two key academic organizations in these areas, have launched a new joint venture to propel data science and to engage and energize our communities to work together.

ACM and IMS will hold an all-day launch event to address topics such as deep learning, reinforcement learning, fairness, and ethics, in addition to discussions about the future of data science and the role of ACM and IMS.

Panels and Panelists

Deep Learning, Reinforcement Learning, and Role of Methods in Data Science

- Shirley Ho, Flatiron Institute
- Sham Kakade, University of Washington
- Suchi Saria, Johns Hopkins University
- Manuela Veloso, J.P. Morgan, Carnegie Mellon University

Robustness and Stability in Data Science

- Aleksander Madry, Massachusetts Institute of Technology
- Xiao-Li Meng, Harvard University
- Richard J. Samworth, University of Cambridge, Alan Turing Institute
- Bin Yu, University of California, Berkeley

Fairness and Ethics in Data Science

- Alexandra Chouldechova, Carnegie Mellon University
- Andrew Gelman, Columbia University
- Kristian Lum, HRDAG (Human Rights Data Analysis Group)

Future of Data Science

- Michael I. Jordan, University of California, Berkeley
- Adrian Smith, Alan Turing Institute

Seating is limited, so register early!
https://www.acm.org/data-science-summit

Keynote Speakers

Jeffrey Dean
Google

David Donoho
Stanford University

Daphne Koller
insitro
Stanford University

DOI:10.1145/3328904

Vinton G. Cerf

Back to the Future

First, allow me to congratulate all the ACM honorees that receive their well-deserved awards this month at the ACM Awards Gala in San Francisco. For an account of the awards this year, please read ACM President Cherri Pancake's summary on p. 5 of this issue.

I want to take you back to the mid-1800s, as the telegraph emerged as a nearly fast-as-light communication technology. You can imagine the excitement when in 1844 Samuel Morse sent his first message between Washington, D.C., and Baltimore, MD. Now you could send messages faster than even a speeding train. If the bad guy robbed a bank and jumped on a train to escape, you could signal the next station to have the police ready to nab the miscreant before the train arrived. The successful laying of a trans-Atlantic cable in 1866 (earlier trials failed in short order) was another major milestone. Then, in 1901, Guglielmo Marconi came along and did it without wires!

These systems worked by "store and forward" since telegrams were sent from station to station, being manually copied and retransmitted "hop by hop." Eventually there were paper tape teletypes that would punch out a tape with the message characters encoded with 5-bit Baudot codes for each letter. The transmitting teletype read the tape and sent the characters to a receiving teletype that would punch out a duplicate tape. The operator would hang the tape on a peg next to the machine that would be used to forward this message to the next hop.[a] There is a wonderful book entitled *The Victorian Internet* by Tom Standage[b] that outlines the history of the telegraph.

Eventually, circuit-switching systems derived from the telephone network could be used to connect the source and destination teletypes directly to each other without the need for intermediate hops, just as voice calls are made. A circuit was set up and the sending teletype would transmit its paper tape and the receiving teletype would punch it out at the other end.

Ironically, the packet switching of the Arpanet reintroduced the store-and-forward method for intercomputer communication. Dedicated circuits connected the packet switches just as the old telegraph sets were connected. When a packet was received, the receiving packet switch examined it and if the packet was not destined for a locally connected computer, it was stored briefly until it reached the head of the line in a queue whereupon it was then forwarded to the next hop (packet switch) along a path to the destination.

The packet switching of the Arpanet reintroduced the store-and-forward method for intercomputer communication.

This was a much faster process than the old manual telegraph method and the forwarding of the packets allowed the concurrent sharing/multiplexing of the dedicated telephone circuit between the packet switches. There was no waiting to set up a dialed circuit. The same circuit could carry many packets going to many destinations without setting up and tearing down circuits. Because all the traffic was split into packets, long files would be easily mixed in with other traffic, reducing the latency for access to the common communication network. With increasingly fast dedicated circuits, the latencies end-to-end dropped and capacity went up leading to the streaming audio, video, and interactive videoconferencing and gaming so prevalent today.

In March 2019 issue of *Communications* there is an important article by Pamela Zave and Jennifer Rexford[c] that reenvisioned the current Internet as a recursively layered network of networks of networks (so to speak) that captures the evolved architecture now manifest. We have come a long way since the 1844 introduction of the telegraph and the 1983 activation of the Internet and there is strong evidence that further evolution is to be expected as new technologies arrive to spark imagination and challenge engineers to improve on the past. C

[a] This was sometimes called "torn tape" telecommunication because you would tear the tape off the receiving teletype.

[b] Standage, T. *The Victorian Internet*. Walker and Company, 1998.

[c] Zave, P.A. and Rexford, J. The compositional architecture of the Internet. *Commun. ACM 62*, 3 (Mar. 2019), 78–87.

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

BLOG@CACM

The *Communications* Web site, http://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:10.1145/3323684 http://cacm.acm.org/blogs/blog-cacm

Is CS Really for All, and Defending Democracy in Cyberspace

Mark Guzdial mulls the difficulty of getting into a computer science class, while John Arquilla ponders political warfare in cyberspace.

Mark Guzdial The Growing Tension Between Undergraduate and K–12: Is CS for All, or Just Those Who Get Past the Caps?

February 3, 2019

http://bit.ly/2HQZhQe

The New York Times recently ran an article titled "The Hard Part of Computer Science? Getting Into Class" (https://nyti.ms/2VaWcNR) about the dramatic increase in undergraduate enrollment, and the inability of U.S. computer science (CS) departments to keep pace with the demand. These facts aren't a surprise. The Computing Research Association report "Generation CS" (https://cra.org/data/generation-cs/) described the doubling and tripling of CS undergraduate enrollment at U.S. institutions from 2006 to 2015. American academia took notice with the 2017 National Academies report on the rapid growth of CS enrollments (http://bit.ly/2CWttnt).

Everyone is trying to figure out how to increase capacity in undergraduate computer science education. CRA-E maintains a list of successful practices for scaling capacity in CS enrollment, many of which were funded by Google (see http://bit.ly/2FUpIBd). *The New York Times* article describes how CS departments are responding to the greater demand than supply in CS classes. We are seeing caps on enrollment, GPA requirements, rations, and even lotteries to allocate the scarce resource of a seat in a CS class.

We may be approaching an inflection point in computing education—and maybe it's one we've seen before. Eric Roberts of Stanford has written a history of undergraduate CS enrollments dating back over 30 years (https://stanford.io/2CNWa7f). He suggests the downturn in enrollment in the late 1980s may have been the result of CS departments' inability to manage rising CS enrollments in the early 1980s. Then, as now, caps and limits were put into place, which sent the message that computer science wasn't for everyone, that only elite students could succeed in computer science. Eric writes, at https://stanford.io/2ODJ4OK:

> *The imposition of GPA thresholds and other strategies to reduce enrollment led naturally to a change in how students perceived computer science. In the 1970s, students were welcomed eagerly into this new and exciting field. Around 1984, everything changed. Instead of welcoming students, departments began trying to push them away. Students got that message and concluded that they weren't wanted. Over the next few years, the idea that computer science was competitive and unwelcoming became widespread and started to have an impact even at institutions that had not imposed limitations on the major.*

Unlike the 1980s, we now have a national movement in the U.S. that wants "CS for All" (https://www.csforall.org/). Primary and secondary schools are increasing access to CS classes. States and school districts are mandating computer science for all students.

We are facing a capacity crunch in undergraduate CS classes, and we are not even close to CS for *all*. While an increasing number of U.S. schools are offering CS classes, only a small percentage of students are taking them up on the offer. Data coming out of U.S. states suggests that less than 5% of U.S. high school students take any computer science, for example, less than 1% in Georgia or Indiana (see state reports at http://bit.ly/2Uk3QZ9). What happens to undergraduate CS enrollment if we get up to 10% of high school students taking computer science, and even a small percentage of those students decide they want to take post-secondary computer science classes? What if we get past 50%?

I don't have a prediction for what happens next. I don't know if we've ever had this kind of tension in American education. On the one hand, we have a well-funded, industry-supported effort to get CS into every primary and secondary school in the U.S. (https://code.org/about/donors). Some of those kids are going to want more CS in college or university. On the other hand, we see post-secondary schools putting the brakes on rising enrollment. Community colleges and non-traditional post-secondary education may take up some of the demand, but they probably can't grow exponentially either. Like the 1980s, CS departments have no more resources to manage growing enrollment—but there is even more pressure than in the 1980s to increase capacity.

The greatest loss in the growing demand for CS classes is not that there will be a narrower path for K–12 students to become professional software developers. As the Generation CS report (http://bit.ly/2Udzecn) showed, a big chunk of the demand for seats in CS courses is coming from CS minors and from non-CS majors. More and more people are discovering that computer science is useful, in whatever career they pursue. Those are the people who are losing out on seats. Maybe they first saw programming in K–12 and now want some more. That's the biggest cost of the capacity crisis. In the long run, increasing computational literacy and sophistication across society could have even bigger impact than producing more professional programmers.

Inability to meet the demand for seats in CS classes may limit the growth in our computing labor force. It may also limit the growth of computational scientists, engineers, journalists, and teachers—in short, a computationally literate society.

Comments

It strikes me that nontraditional learning may be able to take up some of the slack. That won't address the desire for conventional credentialing. I am not certain how that serves folks preparing themselves for non-CS disciplines in which some computation grounding/experience is sought.

Just the same, I wonder if the current control of the spigot by traditional post-secondary arrangements is part of the problem now, and also later if the "demand" decreases for whatever reasons. Having excess capacity on hand, and some way to redirect it, is not the kind of resiliency we afford educational institutions.

—Dennis Hamilton

John Arquilla In (Virtual) Defense of Democracy

March 19, 2019
http://bit.ly/2U9mtj6

In February, *The New York Times* reported that disruptive cyber operations were launched against the Russia-based Internet Research Agency during the 2018 elections in the U.S. These operations took two forms: direct action causing brief shutdowns, and messages to suspected malefactors that sought to deter. The intended goal of these actions was to "protect American democracy."

Neither form of action will prove effective over time. Election propaganda-by-troll can come from myriad sources and surrogates, easily outflanking clumsy efforts to establish some sort of "information blockade." As to deterrence, this is an old chestnut of the age of nation-states. Hacker networks will almost surely *not* be intimidated, whether they are working on their own or at the behest of a malign third party. Indeed, in the future, election hackers are far more likely to ramp up efforts to shape electoral discourses and outcomes—in democracies everywhere.

How, then, can this threat be appropriately countered? There are two ways—to date, neither of which has been chosen. The first has to do with seeking, via the United Nations, an "international code of conduct" (ICC) in cyberspace that would impose behavior-based constraints on both infrastructure attacks and "political warfare." Ironically, it is the Russians who have been proposing an ICC for more than 20 years now—while the American position has been in firm opposition—beginning shortly after the first meeting between U.S. and Russian cyber teams. I co-chaired that meeting, and thought the Russians had proposed a reasonable idea: creating a voluntary arms control regime, like the chemical and biological weapons conventions. It is well past time to return to this important idea.

The other way for democracies to take the sting out of political warfare waged from cyberspace is to clean up their own practices, which in too many countries have descended into outrageous spirals of distortion and lying. What foreign actors are doing pales next to what is being done by the very political parties and citizens of democratic nations now crying "foul" because some *other* is in the game. The world should look to America's Ronald Reagan, who back in the 1980s waged some of the cleanest political campaigns in memory. It will not be easy to stop individuals from becoming bad political actors in cyberspace, but the major political parties should set an example—and an implied moral norm—by rising to the challenge of focusing on fact- and issue-based election campaigns.

One last thought: the U.S. has to be careful about condemning others for engaging in interventions into its political processes. As Dov Levin pointed out in a study conducted while he was a postdoctoral fellow at Carnegie Mellon, from 1946–2000 the U.S. intervened in 81 foreign elections. The number for Russia over the same period was 36. Some have defended American actions by saying that it is okay to intervene when your goal is to shore up liberal forces against authoritarians. But this kind of reasoning can be used by those who attempted to influence the 2016 presidential election in the U.S.; they can say that by "outing" the Democratic Party's backroom efforts to undermine Senator Bernie Sanders' campaign, they were serving the true foundation of democracy: free and fair processes.

Political discourse in cyberspace is a fact of life now, and it will remain so for the foreseeable future in democratic nations. There are two ways to proceed, if the trolls are to be tamed. One involves multilateral action via the United Nations; the other demands an inward-looking devotion—among the political class and at the individual level—to cultivating the better angels of our cyber natures. Both are worth pursuing. C

Mark Guzdial is a professor in the Computer Science & Engineering Division of the University of Michigan. **John Arquilla** is Distinguished Professor of Defense Analysis at the United States Naval Postgraduate School; the views expressed are his alone.

Turing Profile | DOI:10.1145/3323872

Neil Savage

Neural Net Worth

Yoshua Bengio, Geoffrey Hinton, and Yann LeCun this month will receive the 2018 ACM A.M. Turing Award for conceptual and engineering breakthroughs that have made deep neural networks a critical component of computing.

When Geoffrey Hinton started doing graduate student work on artificial intelligence at the University of Edinburgh in 1972, the idea that it could be achieved using neural networks that mimicked the human brain was in disrepute. Computer scientists Marvin Minsky and Seymour Papert had published a book in 1969 on Perceptrons, an early attempt at building a neural net, and it left people in the field with the impression that such devices were nonsense.

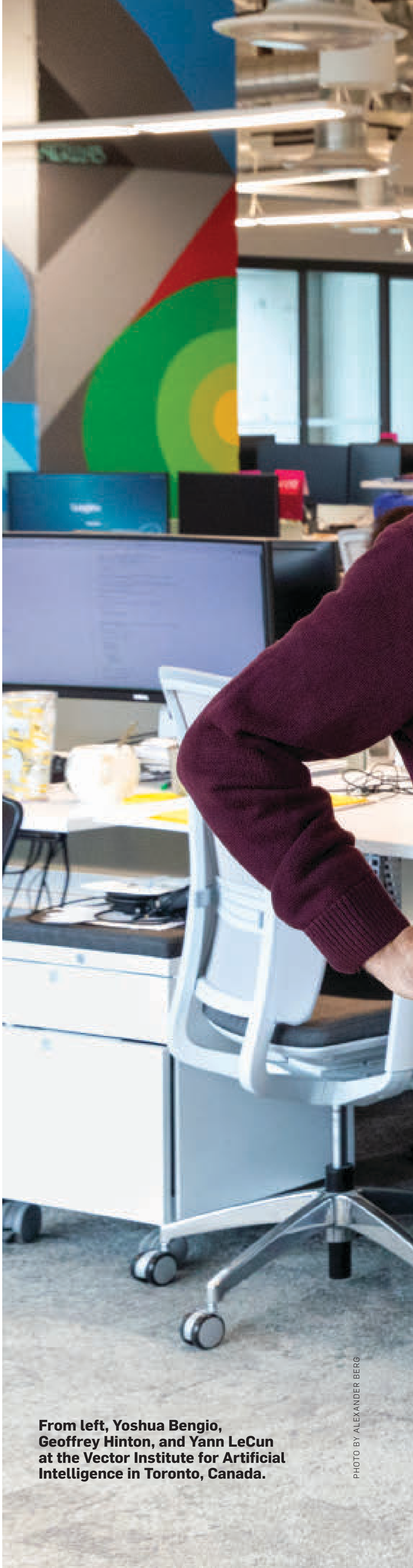
"It didn't actually say that, but that's how the community interpreted the book," says Hinton who, along with Yoshua Bengio and Yann LeCun, will receive the 2018 ACM A.M. Turing award for their work that led deep neural networks to become an important component of today's computing. "People thought I was just completely crazy to be working on neural nets."

Even in the 1980s, when Bengio and LeCun entered graduate school, neural nets were not seen as promising. Many people thought that building a network with random connections across multiple layers, giving it some data, and letting it figure out how to reach the right answer was just asking too much. "People were very suspicious of the idea you could just learn from the data," says Hinton, a professor emeritus at the University of Toronto and now an engineering fellow at Google.

LeCun read Hinton's work including, he says, a paper written in coded language to get around the taboo about neural nets. "I learned about Geoff's existence, and realized this was the man I needed to meet," he says. LeCun did a postdoctoral fellowship in Hinton's lab, then moved to Bell Labs. He's now a professor at New York University (NYU) and director of AI research at Facebook.

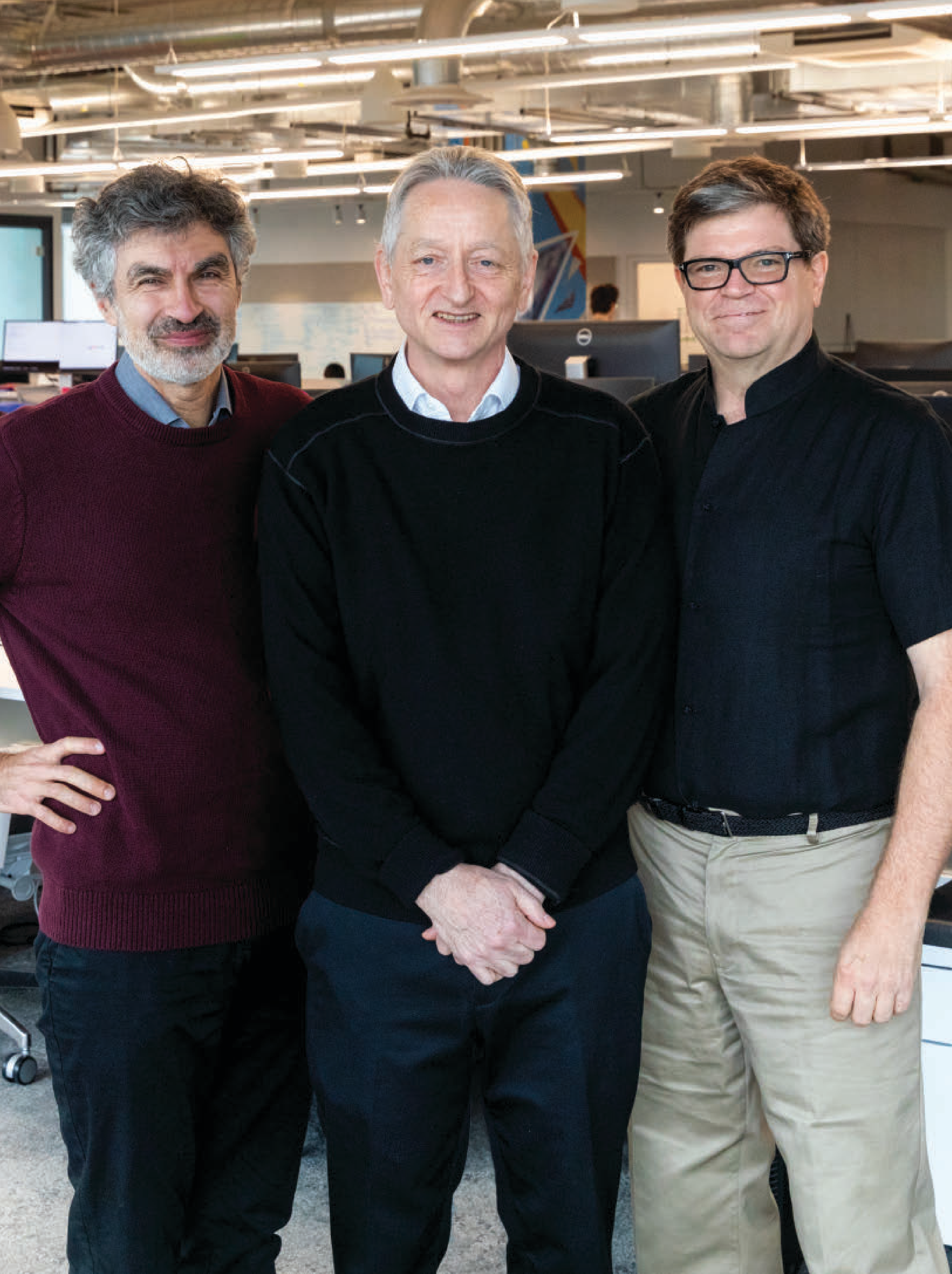
Bengio also wound up at Bell Labs in the early 1990s, where he and Lecun worked together. "What really appealed to me was the notion that by studying neural nets, I was studying something that would be fairly general about intelligence, that would explain our intelligence and allow us to build intelligent machines," Bengio recalls. Today, he is a professor at the University of Montreal, scientific director of Mila (the Montreal Institute for Learning Algorithms), and an advisor to Microsoft.

Their work gained wide mainstream



From left, Yoshua Bengio, Geoffrey Hinton, and Yann LeCun at the Vector Institute for Artificial Intelligence in Toronto, Canada.

PHOTO BY ALEXANDER BERG



acceptance in 2012, after Hinton and two students used deep neural nets to win the ImageNet challenge, identifying objects in a set of photos at a rate far better than that of any of their competitors. Since then, the field has embraced the technology, which has also seen breakthroughs in speech recognition and natural language processing, and could help make self-driving vehicles more reliable.

LeCun says theories about why neural nets would not work—that the training algorithms would get stuck in the extreme values of mathematical functions known as local minima—fell to real-world experience. "In the end, what people were convinced by were not theorems; they were experimental results," he says. Even though there were local minima, those bad enough for an optimization algorithm to get stuck were relatively rare. It turned out that if the neural nets were just big enough for the problem they were trying to solve, they could get stuck, but if they were larger, they became more efficient at optimization. "You make those networks bigger and bigger and they work better and better," LeCun says.

Working both together and independently, the three made important contributions to neural networks. Among their several discoveries, Hinton helped to develop backpropagation, an algorithm that calculates error at the output of the network and propagates the results backward toward the input, allowing the machine to improve its accuracy. LeCun developed convolutional neural networks, which replicate feature detectors across space and are more efficient for image and speech recognition.

Another development that helps the system learn more effectively involves randomly turning off some of the neurons about half of the time, introducing some noise into the network. Bengio says there is noise and randomness in the way living neurons spike, and something about that makes the system better at dealing with variations in input patterns, which is key to making the system useful. "You want to be good at doing the things you haven't yet seen, things that might be somewhat different from the training data," Hinton says.

"Machines are still very, very stupid," LeCun says. "The smartest AI systems today have less common sense than a house cat."

Bengio came up with word embeddings, patterns of neuron activation that represent word symbols, thereby expanding exponentially the system's ability to express meanings and making it possible to process text and translate it from one language to another. Hinton explains that the embeddings make it easier for the system to reason by analogy, rather than by following a logical set of rules; he believes that is more like how the human brain works. The brain evolved to use patterns of neural activity to perform perception and movement, and that makes it more suited to reasoning by analogy rather than logic, he argues.

In fact, artificial intelligence remains limited compared to human intelligence. "Machines are still very, very stupid," LeCun says. "The smartest AI systems today have less common sense than a house cat." Though they excel at recognizing patterns, neural networks have no knowledge of how the world works, and computer scientists have not yet figured out how to give it to them. Humans learn to generalize from a very small number of samples, while neural networks require vast sets of training data. In fact, Hinton says, it was the growth in available datasets, along with faster processors, that led to the "phase shift" from neural networks being a curiosity to a practical approach.

There are hundreds of useful tasks neural networks can accomplish just by using their current pattern recognition capabilities, Hinton says, from predicting earthquake aftershocks to offering better medical diagnoses on the basis of hundreds of thousands of examples. But to give machines a more general intelligence that could solve different types of problems or accomplish multiple tasks will require scientists to come up with new concepts about how learning works, Bengio says. "It might take a very long time before we reach human-level AI," he says.

Meanwhile, society has to have more discussion about how to use artificial intelligence appropriately. Hinton worries about how autonomous intelligent weapons systems might be misused, for instance. LeCun says that without adequate political and legal protections, governments could use the systems to track people and try to control their behavior, or corporations might rely on AI to make decisions but ignore bias in their algorithms.

To address some of these worries, Bengio took part in a group that last December issued the Montreal Declaration for a Responsible Development of Artificial Intelligence, which outlines principles that they say should be used in pushing the technology forward. "We're building stronger and stronger technology based on the premises of science, but the organization of society and their collective wisdom isn't keeping up fast enough. The solution may not be in some new theorem or some new algorithm," he says.

With such concerns in mind, Hinton says he will donate a portion of his share of the $1-million Turing Award prize money to the humanities at the University of Toronto. "If we have science without the humanities to help guide the political process, then we're all in trouble," he says. LeCun says he will likely make a donation to NYU, and Bengio says he's considering some environmental causes.

Based on their experiences as academic heretics who turned out to be right, they advise young computer scientists to stick to their convictions. "If someone tells you your intuitions are wrong, there are two possibilities," Hinton says. "One is you have bad intuitions, in which case it doesn't matter what you do, and the other is you have good intuitions, in which case you should follow them." C

Neil Savage is a science and technology writer based in Lowell, MA, USA.

Science | DOI:10.1145/3323685

Gary Anthes

Lifelong Learning in Artificial Neural Networks

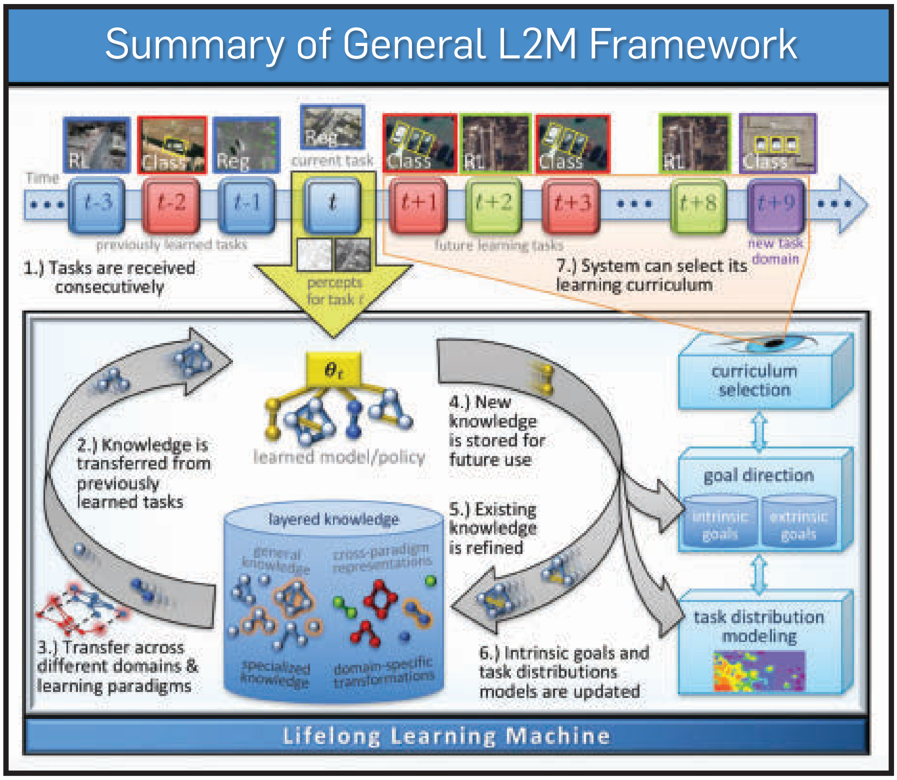
New methods enable systems to rapidly, continuously adapt.

Over the past decade, artificial intelligence (AI) based on machine learning has reached breakthrough levels of performance, often approaching and sometimes exceeding the abilities of human experts. Examples include image recognition, language translation, and performance in the game of Go.

These applications employ large artificial neural networks, in which nodes are linked by millions of weighted interconnections. They mimic the structure and workings of living brains, except in one key respect—they don't learn over time, as animals do. Once designed, programmed, and trained by developers, they do not adapt to new data or new tasks without being retrained, often a very time-consuming task.

Real-time adaptability by AI systems has become a hot topic in research. For example, computer scientists at Uber Technologies last year published a paper that describes a method for introducing "plasticity" in neural networks. In several test applications, including image recognition and maze exploration, the researchers showed that previously trained neural networks could adapt to new situations quickly and efficiently without undergoing additional training.

"The usual method with neural networks is to train them slowly, with many examples; in the millions or hundreds of millions," says Thomas Miconi, the lead author of the Uber paper and a computational neuroscientist at Uber. "But that's not the way we work. We learn fast, often from a single exposure, to a new situation or stimulus. With synaptic plasticity, the connections in our brains change automatically, allowing us to form memories very quickly."


Summary of General L2M Framework
RL
Class
Reg
Reg
current task
Class
RL
Class
RL
Class
Time
t-3
t-2
t-1
t
t+1
t+2
t+3
t+8
t+9
previously learned tasks
future learning tasks
new task domain
1.) Tasks are received consecutively
percepts for task t
7.) System can select its learning curriculum
θ_t
learned model/policy
2.) Knowledge is transferred from previously learned tasks
4.) New knowledge is stored for future use
curriculum selection
goal direction
intrinsic goals
extrinsic goals
5.) Existing knowledge is refined
layered knowledge
general knowledge
cross-paradigm representations
specialized knowledge
domain-specific transformations
3.) Transfer across different domains & learning paradigms
6.) Intrinsic goals and task distributions models are updated
task distribution modeling
Lifelong Learning Machine

The DARPA Lifelong Learning Machines (L2M) Program seeks to develop learning systems that continuously improve with additional experience, and rapidly adapt to new conditions and dynamic environments.

INFOGRAPHIC COURTESY OF DARPA'S ELECTRONICS RESURGENCE INITIATIVE

For more than 60 years, neural networks have been built from interconnected nodes whose pair-wise strength of connection is determined by weights, generally fixed by training with labeled examples. This training is most often done via a method called backpropagation, in which the system calculates an error at the synaptic output and distributes it backward throughout the networks layers. Most deep learning systems today, including Miconi's test systems, use backpropagation via gradient descent, an optimization technique.

> **"In a few years, much of what we consider AI today won't be considered AI without lifelong learning."**

Using that as a starting point, Miconi employs an idea called Hebbian learning, introduced in 1949 by neuro-psychologist Donald Hebb, who observed that two neurons that fire repeatedly across a synapse strengthen their connection over time. It is often summarized as, "Neurons that fire together, wire together."

With this "Hebbian plasticity,"

networks employ a kind of "metalearning"—in essence, they learn how to learn—based on three conceptually simple parameters. Pairs of neurons have the traditional fixed weights established during the training of the system. They also have a plastic weight called a Hebbian trace, which varies during a lifetime according to the actual data it encounters. These Hebbian traces can be computed in different ways, but in a simple example it is the running average of the product of pre- and post-synaptic activity.

The Hebbian traces are themselves weighted by a third fixed parameter, called the plasticity coefficient. Thus, at any moment, the total effective weight of the connection between two neurons is the sum of the fixed weight plus the Hebbian trace multiplied by the plasticity coefficient. Depending on the values of these three parameters, the strength of each connection can be completely fixed, completely variable, or anything in between.

"This is important work," says Ziv Bar-Joseph, a computational biologist at Carnegie Mellon University who was not involved in the work at Uber. "They have taken a principle from biology that was well known and shown it can have a positive impact on an artificial neural network." However, it is too early to say whether the method will represent an important advancement in large, mainstream applications of AI, he says.

With most large AI systems today, Bar-Joseph says, "You optimize, and optimize, and optimize, and that's it. If you get new data, you can retrain it, but you are not trying to adapt to new things." For example, he says, a neural net might have been trained to give

DARPA Projects in Lifelong Learning Machines

Columbia University is learning how to build and train self-aware neural networks, systems that can adapt and improve by using internal simulations and knowledge of their own structures.

The University of California, Irvine, is studying the dual memory architecture of the hippocampus and cortex to replay relevant memories in the background, allowing the systems to become more adaptable and predictive while retaining previous learning.

Tufts University is examining an intercellular regeneration mechanism observed in lower animals such as salamanders to create flexible robots capable of adapting to changes in their environment by altering their structures and functions on the fly.

SRI International is developing methods to use environmental signals and their relevant context to represent goals in a fluid way rather than as discrete tasks, enabling AI agents to adapt their behavior on the go.

—Gary Anthes

ACM News

The Trouble with SMS Two-Factor Authentication

Many use SMS two-factor authentication (2FA) on their smartphones to secure their online accounts, but not everyone understands its potential vulnerabilities.

You've probably seen SMS 2FA in action. An online account, upon login, prompts you to receive a second code on your phone via text message. You receive the second code, then enter it to confirm that you are the legitimate user of the account, and not a hacker.

Yet SMS 2FA can be hacked, too.

In late 2018, Amnesty International reported hackers had hijacked 2FA codes and compromised online accounts; malicious actors had recreated the websites of legitimate services to convince users to reveal their 2FA authentication codes.

SIM swapping is used by hackers to gain access to sensitive accounts "protected" by SMS 2FA, which has resulted in hundreds of millions of dollars in cryptocurrency theft. SIM swapping is when a hacker goes into a phone store pretending to be you, and convinces a staff member to port your SIM card information to a phone they own. The hackers then either convince the original owner to fork over login details, using the swapped SIM to intercept the SMS 2FA code sent after logging in, or they attempt to reset account passwords, using the swapped SIM to intercept the code sent to confirm they are the legitimate account owners.

In July 2018, a suspect was arrested for SIM swapping for the first time, according to crypto/blockchain media outlet CoinTelegraph. The perpetrator allegedly stole $5 million in cryptocurrency using the technique.

SMS 2FA has vulnerabilities, but these are not necessarily flaws in how it is designed, says Kaspersky Lab security researcher Vladimir Dashchenko. "In general, 2FA itself is a secure concept. Yet, the ways it is implemented may differ and could have vulnerabilities," he says.

"Codes sent over the Internet almost always have at least some risk of being stolen," says Mark Risher, Google director of product management for counter-abuse and identity services. "Any form of 2FA improves user security over a password alone; however, not all 2FA provides equal protection. Sophisticated attacks can work around some methods of 2FA."

Risher cites SMS-based phishing attacks as one such method. "Despite this, adding a phone number for two-step verification is still recommended if you can't use any other options," he notes.

The good news is there are other options.

One is Google's own Titan Security Key, a physical key developed using the open source security standard FIDO. When you log into Google services, the SMS 2FA code is sent to the security key instead of your phone; the physical security key then is inserted into your phone to complete the verification process. Risher says the firmware in the security keys has been "sealed permanently into a secure element hardware chip at production time and is designed to resist physical attacks aimed at extracting firmware and secret key material."

Another potential solution is Kaspersky's fraud prevention platform, which leverages machine learning and "continuous analysis of hundreds of parameters in real time" to assess if a user is legitimate. Says Daschenko, "During the whole session, [the system] is analyzing the behavioral and biometric data, device reputation, and other nonpersonalized information to detect any signs of abnormal or suspicious behavior."

That is certainly an improvement over relying on SMS 2FA alone.

—Logan Kugler is a freelance technology writer based in Tampa, FL, USA. He has written for over 60 major publications.

highly accurate results when classifying different kinds of automobiles, but when a new kind of car (a Tesla, say) is seen, the system stumbles. "You want it to recognize this new car very quickly, without retraining, which can take days or weeks. Also, how do you know that something new has happened?"

Artificial intelligence systems that learn on the fly are not new. In "neuroevolution," networks update themselves by algorithms that employ a trial-and-error method to achieve a precisely defined objective, such as winning a game of chess. They require no labeled training examples, only definitions of success. "They go only by trial and error," says Uber's Miconi. "It's a powerful, but a very slow, essentially random, process. It would be much better if, when you see a new thing, you get an error signal that tells you in which direction to alter your weights. That's what backpropagation gets you."

Military Apps

Miconi's ideas represent just one of a number of new approaches to self-learning in AI. The U.S. Department of Defense is pursuing the idea of synaptic plasticity as part of a broad family of experimental approaches aimed at making defense systems more accurate, responsive, and safe. The U.S. Defense Advanced Research Projects Agency (DARPA) has established a Lifelong Learning Machines (L2M) program with two major thrusts, one focused on the development of complete systems and their components, and the second on exploring learning mechanisms in biological organisms and translating them into computational processes. The goals are to enable AI systems to "learn and improve during tasks, apply previous skills and knowledge to new situations, incorporate innate system limits, and enhance safety in automated assignments," DARPA says at its website. "We are not looking for incremental improvements, but rather paradigm-changing approaches to machine learning."

Uber's work with Hebbian plasticity is a promising step toward lifelong learning in neural networks, says Hava Siegelmann, founder and manager of DARPA's L2M program and a computer science professor at the University of Massachusetts, Amherst. "We will never be safe in a self-driving car without it," she says. But it is just one of many necessary steps toward that goal. "It's definitely not the end of the story," she says.

There are five "pillars" of lifelong learning as DARPA broadly defines it, and synaptic plasticity falls into the first of these. The pillars are: continuous updating of memory, without catastrophic forgetting; recombinant memory, rearranging and recombining previously learned information toward future behavior; context awareness and context based modulation of system behavior; adoption of new behaviors through internal play, self-awareness, and self-simulations; and safety and security, recognizing whether something is dangerous and changing behavior accordingly, and ensuring security through a combination of strong constraints.

Siegelmann cites smart prostheses as an example of an application of these techniques. She says the control software in an artificial leg could be trained via conventional backpropagation by its maker, then trained to the unique habits and characteristics of its user, and finally enabled to very quickly adapt to a situation it has not seen before, such as an icy sidewalk.

A computational neuroscientist, Siegelmann says lifelong learning has been a goal of AI researchers for many years, but major advancements have only recently become feasible, enabled by advancements in computer power, new theoretical foundations and algorithms, and a better understanding of biology. "In a few years, much of what we call AI today won't be considered AI without lifelong learning," she predicts.

Miconi's team is now working on making learning more dynamic and sophisticated than it is in his test systems so far. One way to do that is to make the plasticity coefficients, now fixed as a design choice, themselves variable over the life of a system. "The plasticity of each connection can be determined at every point by the network itself," he says. Such "neuromodulation" likely occurs in animal brains, he says, and that may be a key step toward the most flexible decision-making by AI systems. C

DARPA's Lifelong Learning Machines program does not seek incremental improvements, "but rather paradigm-changing approaches to machine learning."

Further Reading

Chang, O. and Lipson, H.
Neural Network Quine, Data Science Institute, Columbia University, New York, NY 10027, May 2018 https://arxiv.org/abs/1803.05859v3

Chen, Z. and Liu, B.
Lifelong Machine Learning, Second Edition, *Synthesis Lectures on Artificial Intelligence and Machine Learning*, August 2018 https://www.morganclaypool.com/doi/10.2200/S00832ED1V01Y201802AIM037

Hebb, D.
The Organization of Behavior: A Neuropsychological Theory, New York: Wiley & Sons, 1949 http://s-f-walker.org.uk/pubsebooks/pdfs/The_Organization_of_Behavior-Donald_O._Hebb.pdf

Miconi, T., Clune, J., and Stanley, K.
Differentiable Plasticity: Training Plastic Neural Networks with Backpropagation, *Proceedings of the 35th International Conference on Machine Learning* (ICML 2018), Stockholm, Sweden, PMLR 80, 2018 https://arxiv.org/abs/1804.02464

Miconi, T.
Backpropagation of Hebbian Plasticity for Continual Learning, *NIPS Workshop on Continual Learning*, 2016 https://github.com/ThomasMiconi/LearningToLearnBOHP/blob/master/paper/abstract.pdf

Gary Anthes is a technology writer and editor based in Arlington, VA, USA

Technology | DOI:10.1145/3323703

Don Monroe

And Then, There Were Three

How long can the silicon foundry sector continue to adapt, as physical limits make further shrinkage virtually impossible?

RELENTLESS YEAR-OVER-YEAR IMPROVEMENTS in integrated circuits don't come cheap. For years, these advances have been boosted in part by silicon foundries that invest in new technology by aggregating demand from design companies that don't have factories of their own. As of last summer, however, only one such "pure-play" foundry continues to pursue the latest silicon generation, along with two companies that also make their own chips. The dwindling of suppliers revives the longstanding question of how the industry can adapt as physical limits eventually make further shrinkage impossible (or impossibly expensive).

Still, the story sounds familiar. "Every time people say Moore's Law has finally hit the wall, people come up with new, innovative approaches to get around it," said Willy Shih, Robert and Jane Cizik Professor of Management Practice at Harvard Business School.

The silicon industry has tracked the 1965 observation by Gordon Moore, co-founder and later head of Intel, that transistor counts were doubling every year (later changed to every two years). This exponential growth became enshrined as a "law," which became a collective self-fulfilling prophesy as companies feared losing business if they fell behind its aggressive schedule. Successive generations were labelled by an ever-shrinking distance, currently 7nm, although this designation long ago lost any clear relationship to the transistor's gate length or other features. In the 1990s, Moore's Law became formalized in the National (after 1998, International) Technology Roadmap for Semiconductors, which spelled out what manufacturers, equipment suppliers, and academic researchers would need to do to keep the industry on track.

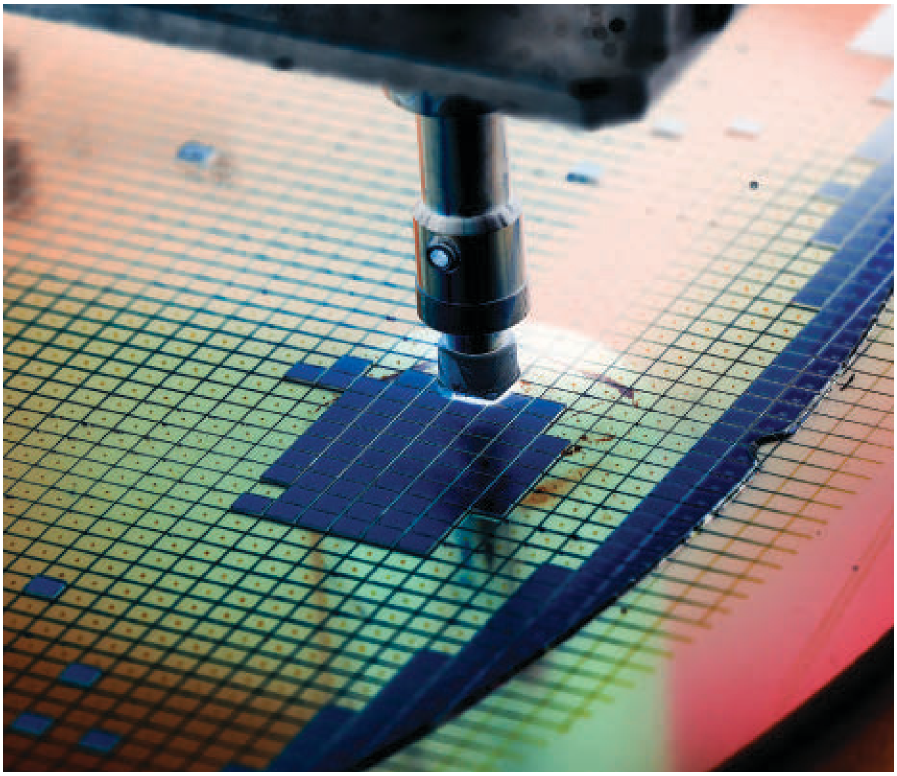


IMAGE BY MACRO PHOTO

Unfortunately, exponentially increasing transistor counts were accompanied by corresponding increases in the costs to build fabrication plants and develop more aggressive processes and novel device structures. These costs, and the need to keep the expensive equipment in constant use, have long made it almost impossible for a smaller company to manufacture a novel chip design itself. "The capital investment to supply a growing market and to push leading-edge research can only be supported by a company that has a large revenue," probably $30 billion a year or more, said Paolo Gargini. "It's just a game for the big boys," said Gargini, formerly at Intel, who has headed the formal roadmap through its recent rebirth as the International Roadmap for Devices and Systems (IRDS).

Nonetheless, when GlobalFoundries announced in August 2018 that it was halting development of its 7nm process, "it was quite a shocker for a lot of people," Shih said. The foundry company had originally projected risk production—early manufacture with relaxed quality guarantees—of 7nm products in spring 2018, and until recently seemed committed. Now, the only remaining pure-play foundry developing leading-edge technology is Taiwan Semiconductor Manufacturing Company (TSMC), whose 7nm process has been in production since early 2018. Besides TSMC, Samsung, which has an important foundry business in addition to manufacturing its own chips, announced in fall 2018 that it was ready for risk production of 7nm. Intel, whose current 10nm process is often regarded as similar to TSMC's 7nm process, devotes most of its attention to its own chips.

Foundries at the Forefront

TSMC pioneered in 1987 the concept of a pure-play foundry. Before that, "If you had a new idea, you really didn't have a place where you could test it" without paying for a dedicated factory, Gargini said. The advent of foundry capacity was "the best thing that could have happened for the industry," he said. "The iPhone would never have existed if we didn't have this model."

At first, TSMC replicated older, less-profitable technologies and grew by "taking the rejects from the leading semiconductor companies." Gargini said. However, "by 2000 or so they were within shooting range of the leading companies."

Later foundries have mostly confined themselves to following the leaders, but GlobalFoundries seemed to have higher aspirations. The company was created in 2009 from the manufacturing operations of Intel's arch-competitor Advanced Micro Devices (AMD). The company also acquired Singapore-based foundry Chartered Semiconductor, and in 2015 added the manufacturing operations of IBM.

Leading-edge semiconductor manufacturing is expensive and challenging, which is one reason AMD and IBM divested that part of their businesses. Into the 1990s, keeping up with Moore's Law could mostly be achieved by "scaling," following rules laid out by IBM's Robert Dennard in 1974 to make better transistors by shrinking lateral dimensions, shrinking layer thicknesses, and increasing doping densities. Packing more transistors on the surface area of a wafer also offered benefits such as reduced cost per transistor, higher speed, and lower power dissipation.

Continued exponential shrinkage brought transistors into collision with fundamental physical limits, though, such as gate oxides just a few atom-layers thick, as well as large leakage currents and other non-idealities in the tiny devices. To sidestep these limits, in the early 2000s manufacturers introduced multiple revolutionary innovations, such as high-dielectric-constant (high-k) gate dielectrics, metal gates, strained silicon, and the nonplanar transistors known as FinFETs.

More innovation will be needed, including in process technology. Especially challenging has been the lithography that prints the circuits, using progressively shorter ultraviolet wavelengths to create tinier features. This shrinkage stalled for years at a wavelength of 193 nm because the next huge jump, to extreme ultraviolet (EUV) at 13.5nm, requires different sources, optics, and exposure techniques. Instead, designers have exploited liquid immersion, multiple exposures, and other tricks to extend 193nm lithography. With the 7nm generation, EUV is finally being used for some processing levels, but economically viable throughput and yield won't come easily.

Consolidation is "troubling" for U.S. semiconductor manufacturing, because "a vast amount of the world's advanced foundry capacity is in TSMC's hands."

Shakeout

These challenges are not new, but the withdrawal of companies from the leading edge raises a "very valid question," Shih said. "If there's less competition, are we going to push the frontier less?" So far, there are still multiple suppliers.

"As long as you have two, it's sufficient; if you have three it's great," Gargini said. "Samsung can do a lot of the stuff that TSMC can do," and TSMC's lead already meant that "there's nothing that is so special that GlobalFoundries was doing," Gargini said. AMD, for example, already made many of its most advanced central processing units (CPUs) and graphics processing units (GPUs) at TSMC.

Still, Shih notes that the consolidation is "troubling" for U.S. semiconductor manufacturing, because "a vast amount of the world's advanced foundry capacity is in TSMC's hands in three fabs in Taiwan." He added that "People who worry about the defense-industrial base are very concerned about this issue."

ACM Member News

SEEKING NEW WAYS TO BUILD AND MAINTAIN OPEN DISTRIBUTED SYSTEMS

Gul Agha is a professor in the Department of Computer Science, and director of the Open Systems Laboratory, at the University of Illinois at Urbana-Champaign.

Agha received his undergraduate degree from the California Institute of Technology. He earned a master's degree in psychology and a Ph.D. in computer and communication science from the University of Michigan at Ann Arbor, but did his dissertation research at the Massachusetts Institute of Technology.

In 1989, he joined the faculty at the University of Illinois, where he has remained ever since. "One of the joys of being an academic," Agha says, "has been my ability for life-long learning and to acquire new knowledge and perspectives."

Agha's research interest is in understanding the nature of concurrent computation, leading to new ways to build and maintain open distributed systems.

"My research has spanned diverse areas such as programming languages, software engineering, cyber-physical systems, and formal methods," Agha says. "I want to develop unifying programming abstractions for new generation of applications, such as IoT (the Internet of Things) for Smart Cities." These applications require concurrency and coordination, notions of approximation and stochastic behavior, and integration with continuous spatiotemporal variables.

A real-world project on which Agha collaborated was the implementation of the world's largest sensor network to continuously monitor the structural health of South Korea's 484-meter Jindo Bridge, connecting the Korean mainland to Jindo Island. The sensor network promises a robust, significantly lower-cost alternative to traditional structural inspection techniques, Agha says.

—John Delaney

To be sure, GlobalFoundries and others (including TSMC) can still build very powerful products using older technologies. Moreover, Shih notes, "Some people say that, once we went below 14nm, or perhaps even higher like 22nm, the unit cost per transistor stopped decreasing and started increasing again." As a result, "more and more users say 'That [leading-edge] process is so expensive, I actually don't need it,'" he said, unless they are "making things for cellphones or FPGAs or the bleeding-edge stuff like Intel microprocessors, where you really need the ultimate in performance and power."

Indeed, a manufacturer that specializes in digital logic may not need a broad range of processes. In contrast, foundries support a whole range of devices, such as image sensors, and devices for analog, radio-frequency, and ultra-low-power circuits. Reliably implementing such mix-and-match processes in a design environment that lets multiple customers use them is often more important to designers than having the latest-generation technology. For example, although TSMC boasts dozens of high-end customers for its 7nm process, for example, it continues to support older-generation processes, even the 180nm technology it introduced 20 years ago, which is good enough for many customers.

If leading-edge development slows down, though, it might give other companies, including those in mainland China, more chance to compete. "The Chinese are having trouble at the leading edge, but they're catching up on some of the trailing-edge technologies," Shih said. "The thing that is driving TSMC is less competition from GlobalFoundries; it's competition from Made in China 2025 [a Chinese program to improve domestic manufacturing competitiveness]."

A Bright Future?

In the end, though, no amount of innovation can extend exponential scaling forever. Logic designers "are waiting for EUV to save the game," Gargini said, but even if advanced lithography buys a few years, "that solution comes to an end." In perhaps 2020 or 2021, he conjectured, "Samsung, TSMC, or Intel, one of them will make a big announcement that their next product is 3D [three-dimensional]," which would offer more transistors through vertical stacking. Memory manufacturers (including Samsung) have already begun to introduce 3D structures, both by stacking processed layers and growing multiple layers of devices (see "Electronics are Leaving the Plane," *Communications*, August 2018). Memory has special advantages for 3D structures, such as uniform and redundant layouts, and low power (because most transistors are idle).

In contrast, in logic applications, many more transistors are active, and removing the heat they produce is enormously challenging even in the easier-to-cool planar layout. So far, logic companies are testing the 3D waters with advanced packaging techniques for GPUs and other high-performance products. "We still can squeeze another two or three generations out of 2D," Gargini said, but he sees full 3D as inevitable and adding another 15 years of performance growth. "3D is not really as much of a revolution" or as risky as the process innovations the industry has already implemented, he said. "The big guys can do it anytime they decide to do it."

The semiconductor industry faces challenges that we may look back on as the end of Moore's Law. Nonetheless, there are continued opportunities for better products, and so far there are still foundry companies ready and able to enable new designs. "There is a bright future," Gargini insists. "I think it's a very good balance."

Further Reading

International Roadmap for Devices and Systems 2017 Edition, IEEE, https://irds.ieee.org/roadmap-2017

Shih, W. C., Chien, C.F., Shih, C., and Chang, J.
The TSMC Way: Meeting Customer Needs at Taiwan Semiconductor Manufacturing Co., Harvard Business School Case Collection 610-003, August 2009, https://www.hbs.edu/faculty/Pages/item.aspx?num=37868

Monroe, D.
Electronics are Leaving the Plane, *Communications*, August 2018, https://cacm.acm.org/magazines/2018/8/229776-electronics-are-leaving-the-plane/fulltext

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

Milestones

ACM, CSTA Announce Cutler-Bell Prize Winners

ACM and the Computer Science Teachers Association (CSTA) will bestow the 2018–2019 Cutler-Bell Prize promoting computer science and empowering students to pursue computing challenges beyond the classroom upon four graduating high school students.

Each will receive a $10,000 cash prize toward tuition at the institution they will attend next year.

The winning projects illustrate the diverse applications being developed by the next generation of computer scientists:

NAVEEN DURVASULA, SILVER SPRING, MD
Durvasula developed a method to predict, for a given patient-donor pair, the expected quality and waiting time of the transplant they would receive through kidney exchange.

ISHA PURI, CHAPPAQUA, NY
Puri focused on development of a system to detect the direction and frequency of gaze fixation to test for and diagnose dyslexia.

ESHIKA SAXENA, BELLEVUE, WA
Saxena developed the "HemaCam," a clip-on attachment that turns a smartphone camera into a microscope capable of capturing blood cell images for disease screening.

VARUN SHENOY, CUPERTINO, CA
Shenoy created an effective method to diagnose the onset of wound complications during surgical operations.

Said ACM president Cherri M. Pancake. "These are the kinds of skills students will increasingly need in our digital age. In short, the Cutler-Bell Prize encourages students to see the possibilities, as well as the excitement, that computing offers."

Added CSTA executive director Jake Baskin, "Our winners have created projects that have applicable real-world solutions, all resulting from the high-quality computer science education they have received."

Society | DOI:10.1145/3323702

Keith Kirkpatrick

Ethics in Technology Jobs

Employees are increasingly challenging technology companies on their ethical choices.

ORGANIZED PROTESTS AGAINST companies are hardly a new phenomenon, as people have boycotted or protested both corporate policies and actions for years. For example, a global protest of international agrochemical and agricultural biotechnology corporation Monsanto in 2013 saw coordinated marches across 52 countries and 436 cities. In 2010, thousands of people in the U.S. protested against oil giant BP for its role in the Deepwater Horizon oil spill. And in the late 1990s, U.S. gun owners protested against gun manufacturers Colt Manufacturing Company and Smith & Wesson for their perceived cooperation with then-President Bill Clinton's gun control efforts.

Yet many of the corporate protests that have occurred against technology companies over the past year were marked by a distinct difference: they were often organized by, led, or coordinated with workers at the very companies being protested. The impetus for these walkouts appears to be largely two issues: the presence of a culture of inequality at technology companies, and the use of technology for what workers consider to be unethical or harmful activities.

Although there is precedent for tech workers protesting against their employers, such as when defense workers in the 1980s pushed back against their employers' participation in the development of the Strategic Defense Initiative, colloquially known as Star Wars, the difference is that tech workers feel more empowered to speak out today.

"[Workers] actually see that their words and action can have a real impact on a broader scale," says Mehran Sahami, a professor of computer science at Stanford University. Sahami points to the success former Uber employee Susan Fowler had with blog posts she wrote that detailed a culture of sexual harassment at the ride-sharing giant, which ultimately led to changes at the company and the dismissal of its former CEO, Travis Kalanick. "Fowler's actions showed that even individual tech workers, by speaking up, can actually have a large effect on the organization that they're in or were formerly in," Sahami says.

> "Silicon Valley companies lead the way in … science and technology, but when it comes to issues of privacy, creating inclusive workplaces, and ethics, they seem to be devolving."

It is not just a culture of misogyny that is irritating workers and spurring them into action; a lack of transparency is also a key catalyst for workers to band together to make their feelings known. One example was Google's handling of a $90-million exit payment to Andy Rubin, a key executive of the company and the creator of the Android mobile operating system. Upon Rubin's departure from the company in 2014, Google failed to disclose it had received a complaint that Rubin had committed an act of sexual misconduct against another employee, and that an investigation had confirmed its veracity. In October 2018, a report in *The New York Times* made these details public.

Upon that disclosure, Google CEO Sundar Pichai sent a memo to staff noting that the company had taken "an increasingly hard line" on inappropriate conduct at work and had fired 48 people, including 13 senior managers, in the previous two years, without giving any of them exit packages. Just prior to a November 1 protest by employees known as "The Walkout for Real Change," Pichai sent out a follow-up note apologizing "for the past actions and the pain they have caused employees" and indicating that employees would be supported if they protested.

Despite the apology, thousands of Google employees around the world walked out on November 1, and organizers issued a statement demanding more transparency from Google around its handling of sexual harassment, an end to pay and opportunity inequality, and more employee empowerment overall. In addition, the group requested that an employee representative be appointed to the company's board and that Google end "forced arbitration" in cases of harassment and discrimination, a practice that prevents employees from taking cases to court.

"Silicon Valley companies lead the way in the fields of science of and technology, but when it comes to issues of privacy, creating inclusive workplaces, and ethics, they seem to be devolving," says Congresswoman Jackie Speier, who represents San Francisco and parts of Silicon Valley, and publicly supported the walkouts.

Lack of diversity is a problem in the tech industry. For example, nearly 70% of Google employees are men and 53% are non-Hispanic whites, according to the Google Diversity Annual Report 2018. Among leadership roles, the numbers within Google are even less diverse, as 67% are white non-Hispanic and 75% are men.

"On the issue of diversity, I continue to hear from women and other workers in the tech industry who are harassed, bullied, assaulted, and ignored because they weren't frat buddies with the CEO or turned down sexual overtures," Speier says. "It's a cultural crisis, and as I've made clear to the tech companies in and around my district, the industry will never reach its full potential until this crisis is addressed."

Google is hardly the only company being subjected to protests from its own employees; others also have protested how technology being developed by the companies they work for is being used by government entities. Representatives from Amazon, Salesforce, and Microsoft signed petitions and held demonstrations objecting to how their work is being used for surveillance, or to separate families at the U.S. border. According to Leigh Hafrey, a Senior Lecturer at the Massachusetts Institute of Technology Sloan School of Management and author of the book *The Story of Success: Five Steps to Mastering Ethics in Business*, these protest actions are occurring because workers are more aware of questions of social justice and what constitutes appropriate and inappropriate behavior.

"We've had a lot of social movement over the past several decades that raised awareness and made people conscious of what can potentially happen within organizations," Hafrey says.

Indeed, thousands of workers at Amazon, Google, Microsoft, and Salesforce have signed petitions asking their respective management teams to cancel or withdraw from contracts with U.S. government agencies, including Immigration and Customs Enforcement, Customs and Border Protection, and the Department of Defense. The public nature of these protests and petitions may be having an effect; in June 2019, Google employees succeeded in getting the company to agree not to renew its deal to help the Pentagon build artificial intelligence tools for drone warfare.

Other protests have been less than successful. Salesforce.com employees gathered twice in 2018 in front of the company's headquarters in San Francisco to protest the firm's multimillion-dollar contract with the U.S. Customs and Border Protection agency. While CEO Marc Benioff condemned the agency's separation of families at the border, he refused to cancel the contract, and the company still supplies software to the agency, despite continuing pressure from workers.

Ultimately, workers may be able to make their voices heard, but management at many large companies are likely to be more focused on how their decisions impact the company's bottom line, and so may not always bow to the wishes of employees.

Ceren Cubukcu, an employment consultant and author of *Make Your American Dream A Reality: How to Find a Job as an International Student in the U.S.*, says employees may simply decide to work for another company if they have a problem with a technology company's actions, rather than protesting to get their employer to change course.

"In some projects, especially for IT/high tech projects, you don't even know what the whole project will be at the end because you work in teams, and only the top management knows about the whole project," Cubukcu says. "If you don't feel comfortable in your job or don't like your work, you can always try to switch to another job, and the company can always replace you with some other employee."

That said, the bargaining position for many tech workers is perhaps stronger than it ever has been in history, given that programmers, software engineers, and data scientists that are talented, hardworking, and reliable are relatively hard to find and keep.

"Finding good technical people is difficult," Sahami says, "so companies pay more attention to their workers because they realize that these are highly skilled people who are difficult to find. If those tech workers leave, it's going to have a serious impact on the productivity of the company."

Even young people who have yet to establish themselves in their careers are trying to flex their muscles, shunning companies they don't agree with during the interview and hiring process. A *Buzzfeed* article published in August 2018 included several accounts of tech workers that declined lucrative positions at major technology companies because they disagreed with the company's practices or ethical positions, relating to either the products or services the company builds, the customers to which the companies sell, or how the companies treat their own employees.

"Questions have always been raised about what companies do and why they do it," Hafrey says. "We're just seeing it in a way that I think maybe we were not previously considering because we were enamored of the bright future that our recent technologies promised us, and we are now realizing the downside or potential downsides of some of those technologies."

Sahami adds that there may be a generational reason for the increasing level of activism in the technology field. "There's lots of data that shows, for example, that many in the younger generation look for work that they believe that has value and that's more important to them than just the paycheck; it's believing that they're having some sort of social impact," Sahami says.

"There's been a lot of bad behavior, and not just in the tech industry, but more broadly around issues of sexual harassment that has been in some sense tolerated for a long time. And it shouldn't have been tolerated, but over time, culture changes and people are willing to speak up more about that being unacceptable and so, generationally, we begin to call out more and more of these bad behaviors that's been happening and try to rectify it." C

Further Reading

Fowler, S.
Reflecting on one very, very strange year at Uber, Feb. 19, 2017, https://www.susanjfowler.com/blog/2017/2/19/reflecting-on-one-very-strange-year-at-uber

Keller, M., and Larsen, K.
'Enough is enough': Google workers in San Francisco, Mountain View, Sunnyvale walk out in protest of treatment of women, November 1, 2018, ABC 7 News San Francisco, https://abc7news.com/business/enough-is-enough-bay-area-google-workers-walk-out-in-protest/4596806/

Brown D.
"Google Diversity Annual Report 2018." Diversity.Google. https://static.googleusercontent.com/media/diversity.google/en//static/pdf/Google_Diversity_annual_report_2018.pdf

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

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.

acm 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:10.1145/3325279

Linnet Taylor

▸ Michael L. Best, Column Editor

Global Computing
Global Data Justice

A new research challenge for computer science.

When the world's largest biometric population database—India's Aadhaar system—was challenged by activists the country's supreme court issued a historic judgment. It is not acceptable, the court said, to allow commercial firms to request details from population records gathered by government from citizens for purposes of providing representation and care. The court's logic was important because this database had, for a long time, been becoming a point of contact between firms that wanted to conduct ID and credit checks, and government records of who was poor, who was vulnerable, and who was on which type of welfare program. The court also, however, said that this problem of public-private function creep was not sufficiently bad to outweigh the potential good a national population database could do for the poor. Many people, they said, were being cheated out of welfare entitlements because they had no official registration, and this was more unfair than the monetization of their official records.

This judgment epitomizes the problem of global data justice. The databases and analytics that allow previously invisible populations to be seen and represented by authorities, and which make poverty and disadvantage harder to ignore, are a powerful tool for the marginalized and vulnerable to claim their rights and entitlements, and to demand fair representation.[2] This is the claim the United Nations is making[5] in relation to new sources of data such as cellphone location records and social media content: if the right authorities can use them in the right way, they can shine a light on need and deprivation, and can help evaluate progress toward achieving the Sustainable Development Goals. If data technologies are used in a good cause, they confer unprecedented power to make the world a fairer place.

How to set boundaries for powerful international actors is a question yet to be solved in any field.

That 'if', though, deserves some attention. The new data sources' value to the United Nations, to humanitarian actors, and to development and rights organizations are only matched by their market value. If it is possible to monitor who is poor and vulnerable, it is also possible to manipulate and surveil. Surveillance scholar David Lyon[3] has said that all surveillance operates along a spectrum between care and control: a database like Aadhaar can be used to channel welfare to the needy, but it could also be used to target consumers for marketing, voters for political campaigns, transgender people or HIV sufferers for exclusion—the list is endless. The possibilities for monetizing the data of millions of poor and vulnerable people are endless, and may be irresistible if hard boundaries are not set. But how to set boundaries for powerful international actors is a question yet to be solved in any field.

Data technologies have very different effects in different social, eco-

A woman has her eyes scanned while others wait during the Aadhaar registration process in India circa October 2018. Aadhaar produces identification numbers to individuals issued by the Unique Identification Authority of India on behalf of the Government of India for the purpose of establishing the identity of every single person.

PHOTO BY DAVID TALUKDAR/NURPHOTO VIA GETTY IMAGES

nomic, and political environments. WhatsApp, for example, allows parents' groups to message each other about carpooling. It also facilitates ethnic violence in India and Myanmar[a] and facilitates extremist politics[b] in Brazil. Technology almost always has unintended consequences, and given the global reach of apps and services, the consequences of our global data economy are becoming less and less predictable.[1]

Global data justice researchers are aiming to frame new governance solutions that can help with this global level of unpredictability. In this emerging research field, we are exploring how the tools we have are globalizing: regulation, research ethics, professional standards and guidelines are all having to be translated into new environments, and get understood differently in different places. Nigeria, the U.S., and India, for example, will each have a different idea of what is 'good' or 'necessary' to do with data technologies, and how to regulate their development and use. Our research asks how to reconcile those different viewpoints, given that each of those international actors—plus myriad others—will have the power to develop and sell data technologies that will affect people all around the world.

Currently much of the international discussion revolves around harmonizing data protection amongst countries, and getting technology developers to agree on ethical principles and guidelines. Neither of these are bad ideas, but each can go in a radically different direction depending on local views on what is good and desirable. Strongly neoliberal, pro-market countries will develop different principles from more socialist ones, and even if they work from similar templates, will apply them differently. Democracies will set boundaries for data collection and use that are different from those of authoritarian states—yet we all have to work together on this problem. Like climate change, any unregulated data market affects us all.

So neither harmonized data protection nor ethical principles are the answer—or at least not on their own. Ethics, at the moment at least, is too frequently just a cover for self-regulation.[6] We need to ask global questions about global problems, but we are often stuck looking at our own environment and our own set of tools, without understanding what kind of toolkit can address the international-level consequences of our growing data economy.

If we ask this global question, instead: How to draw on approaches that are working in different places,

[a] See https://bit.ly/2zWDIKO
[b] See https://nyti.ms/2EzEP5h

and how to set boundaries and goals collectively for our global data economy?, we arrive at questions about both justice, and intercultural understandings of it. We need not only to be able to articulate principles of justice and fairness, but to have a productive discussion about them with nations that see things very differently.

Research on global data justice[4] is starting from this larger question of how to pick and articulate principles that people seem to agree on around the world; we will then work on how those should be turned into tools for governing data—and creating the institutions we need to do so, if they do not exist. Researchers working on this problem (who now include philosophers, social scientists, lawyers, computer scientists and informatics scholars, doing research in Europe, the U.S., Africa, and Asia) have to try to capture at least three conflicting ideas about what data technologies do and what their value is.

These conflicting ideas offer three main principles: first, that our visibility through data should work for us, not against us. We should be visible through our data when we need to be, in ways that are necessary for our well-being, but that it should be part of a reasonable social contract where we are aware of our visibility and can withdraw it to avoid exploitation. Second, that we should have full autonomy with regard to our use of technology. We should be able to adopt technology that is beneficial for us, but using a smartphone or being connected should not be linked to our ability to exercise our citizenship. Someone who has to use social media to get a national identity document or who has to provide biometrics through a private company in order to register for asylum, is not using data technologies so much as being used by them. Lastly, the duty of preventing data-related discrimination should be held by both individuals and governments. It is not enough to demand transparency so that people can protect themselves from the negative effects of profiling: people should be proactively protected from discrimination by authorities who have the power to control and regulate the use of data.

What is fair or innocuous in one place may be unfair or harmful in another.

These principles form a starting point for understanding how similar challenges play out in different places. The task of research is to identify where common responses to those challenges are emerging, to draw out lessons for governance, and to suggest ways to operationalize them. Translating this vision to the global level is a huge challenge. To do this, we have to place different visions of data's value and risks in relation to each other, and seek common principles that can inform governance. Framing what global data justice might mean involves law, human rights, the anthropology of data use and sharing, the political economy of the data market and of data governance more broadly, and international relations.

This global problem is also becoming part of the agenda of computer science and engineering. The agenda of justice in relation to digitization is under formation, and needs input from all the fields doing conceptual and applied work in relation to the digital. It is not a task any individual field can address on its own, because work on data technology has evolved beyond the point where those who conceptualize and develop systems can understand what effects they will have on the global level. What is fair or innocuous in one place may be unfair or harmful in another.

Data justice should provide a lens through which we can address questions about how to integrate values into technology, but it is a higher-level question that cannot be answered with guidelines or with toolkits for privacy or explainability (despite the importance of these approaches). It is a conceptual question, though it leads to practical questions of governance: we wish to conceptualize how data should be governed to promote freedom and equality. This is not something academia can do on its own, but is a long-term challenge to be addressed in collaboration with policymakers, and in consultation with everyone affected by the data economy.

Computer scientists are already part of this process. When they conceptualize and build systems, they make choices that determine how data gets constructed and used. Understanding how computer scientific research connects to the human and to the social world, and how CS research contributes to particular outcomes, is the first step. Making connections between that understanding and social scientific research is a necessary first step. This process is taking place at some computer scientific conferences (notably ACM FAT*, which is now integrating social science and law tracks), but is also visible in smaller workshops and interdisciplinary programs where social scientists and computer scientists come together to work on the social implications of data science and AI, to publish together and to build a research agenda. This work will grow in scale and importance in the coming years, with the notion of global data justice as a benchmark for the inclusiveness and breadth of the debate. C

References

1. Dencik, L., Hintz, A., and Cable, J. Towards data justice? The ambiguity of anti-surveillance resistance in political activism. *Big Data & Society 3*, 2 (Feb. 2016), 1–12; https://bit.ly/2VxoF0A
2. Heeks, R. and Renken, J. Data Justice For Development: What Would It Mean? (Development Informatics Working Paper Series No. 63). Manchester, U.K., 2016; https://bit.ly/2UKVIRr
3. Lyon, D. *Surveillance Studies: An Overview.* Polity Press, Cambridge, 2007.
4. Taylor, L. What Is Data Justice? The Case for Connecting Digital Rights and Freedoms on the Global Level. Big Data and Society, 2017; https://bit.ly/2uZjxXb
5. United Nations. A World that Counts: Mobilising the Data Revolution for Sustainable Development. New York, 2014; https://bit.ly/1it3l8P
6. Wagner, B. Ethics as an escape from regulation: From ethics-washing to ethics-shopping? In M. Hildebrandt, Ed. *Being Profiled: Cogitas Ergo.* Sum Amsterdam University Press, Amsterdam, 2018, 84–90.

Linnet Taylor (l.e.m.taylor@tilburguniversity.edu) is an associate professor at Tilburg Law School, Tilburg University, The Netherlands.

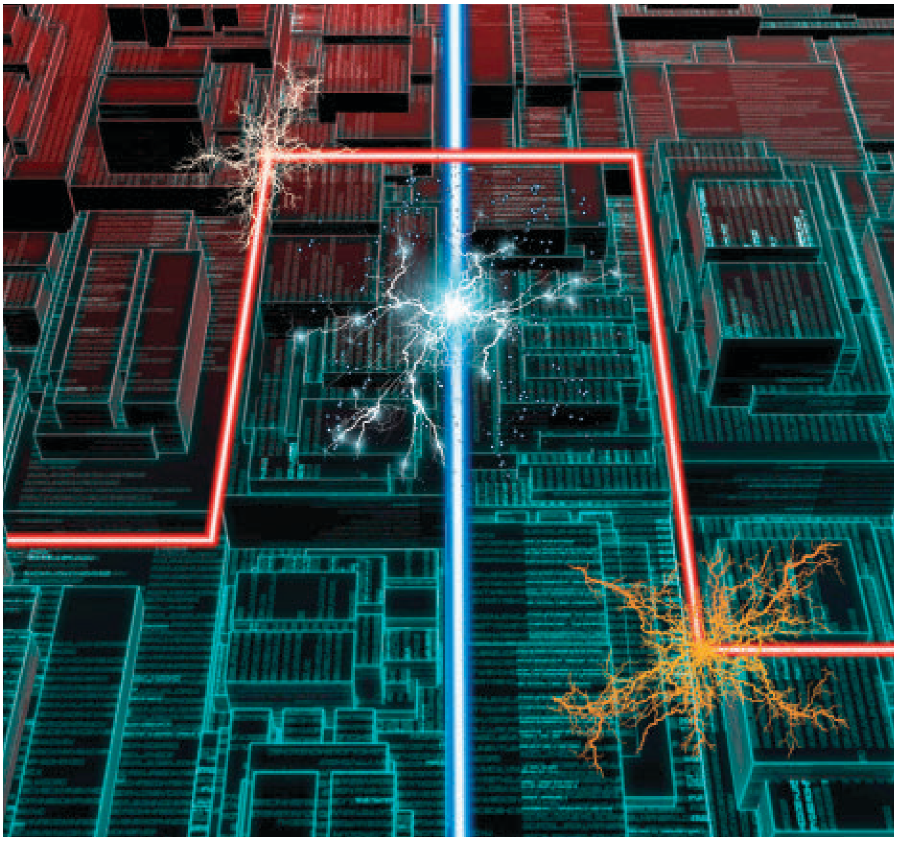
This work is funded by Horizon 2020 ERC Starting Grant #757247 DATAJUSTICE.

DOI:10.1145/3325284

A.T. Markettos, R.N.M. Watson, S.W. Moore, P. Sewell, and P.G. Neumann

Inside Risks
Through Computer Architecture, Darkly

Total-system hardware and microarchitectural issues are becoming increasingly critical.



COLLAGE BY ANDRIJ BORYS ASSOCIATES, USING SHUTTERSTOCK

SPECTRE,[11] MELTDOWN,[13] FORESHADOW,[18,20] Rowhammer,[9] Spoiler,[9]—suddenly it seems as if there is a new and unending stream of vulnerabilities in processors. Previous niche concepts such as speculative execution and cache timing side-channels have taken center stage. Across the whole hardware/software system, new vulnerabilities such as insufficiently protected memory access from untrustworthy PCIe or Thunderbolt USB-C peripherals,[15] malicious Wi-Fi firmware,[4] or alleged hardware implants[14] are also starting to emerge.

We may be facing a crisis in systems design. What might we do about it? Here, we consider whether existing approaches are adequate, and where substantial new work is needed.

Prove, Don't Patch

Many existing commercial operating systems have extensive vulnerabilities. The MITRE repository of common software security vulnerabilities (CVEs: http://cve.mitre.org) currently has over 110,000 open enumerated vulnerabilities that have been reported (excluding ones that have been resolved, and totally ignoring countless other vulnerabilities that have never been reported); the list is growing at a rate of approximately 50 new vulnerabilities each day. Patches cannot possibly keep up with the weaknesses. In addition, patching silicon takes years and potentially costs billions of dollars, which clearly tilts the balance firmly in favor of the attacker.

Recent advances such as the seL4 microkernel,[10] the CertiKOS virtual-machine hierarchy,[8] and the CompCert verified compiler[12] have significantly contributed to the state of the art in formally proven correctness of operating-system kernels. This technology is not yet widespread, but it offers the potential to prove the absence of large classes of attacks. It relies on trustworthy models of the architectural abstraction—the hardware/software interface—and those too have advanced recently, in work by the authors and others.[1,6]

Looking Behind the Hardware Curtain

It has recently become clear that this is not enough, in several ways. First, processor hardware (typically subject to extensive verification) has long been assumed to provide a solid foundation for software, but increasingly suffers from its own vulnerabilities. Second, increasing complexity and the way systems are composed of many hardware/software pieces, from many vendors, means one cannot think just in terms of a single-processor architecture. We need to take a holistic view that acknowledges the complexities of this landscape. Third, and most seriously, these new attacks involved phenomena that cut across the traditional architectural abstractions, which have intentionally only described the envelopes of allowed *functional* behavior of hardware implementations, to allow implementation variation in performance. That flexibility has been essential to hardware performance increases—but the attacks involve subtle information flows via performance properties. They expose the hidden consequences of some of the microarchitectural innovations that have given us ever-faster sequential computation in the last decades, as caching and prediction leads to side-channels.

Hardware Vulnerabilities

Ideally, security must be built from the ground up. How can we solve the problem by building the foundations of secure hardware?

For years, *hardware security* to many people has meant focusing on the physical layers. Power/electromagnetic side-channels and fault injection are common techniques for extracting cryptographic secrets by manipulating the physical implementation of a chip. These are not without effectiveness, but it is notable that the new spate of attacks represents entirely different, and more potent, attack vectors.

One lesson from the physical-layer security community is that implementation is critical. Hardware definition languages (HDLs) are compiled down to connections between library logic cells. The logic cells are then placed and routed and the chip layer designs produced. One tiny slip—at any level from architecture to HDL source and compiler, to cell transistor definitions, routing, power, thermals, electromagnetics, dopant concentrations and crystal lattices—can cause a potentially exploitable malfunction. Unlike the binary code of malware, there is no way to observe many of these physical properties. As a result, systems are more vulnerable to both design mistakes and supply-chain attacks.

Designers need to understand more of what takes place in layers above or below their field of expertise.

As the recent attacks demonstrate, side-channels are becoming more powerful than expected. Traditional physical-layer side-channels are a signals-from-noise problem. If you record enough traces of the power usage, with powerful enough signal processing, you can extract secrets. Architectural side-channels have more bandwidth and better signal-to-noise ratios, leaking much more data more reliably.

If we take a systems-oriented view, what can we say about the problem? First of all, the whole is often worse than the sum of its parts. Systems are composed of disparate components, often sourced from different vendors, and often granting much greater access to resources than needed to fulfill their purpose; this can be a boon for attackers. For example, in Google Project Zero's attack on the Broadcom Wi-Fi chip inside iPhones,[4] the attackers jumped from bad Wi-Fi packets to installing malicious code on the Wi-Fi chip, and then to compromising iOS on the application processor. Their ability to use the Wi-Fi chip as a springboard multiplied their efficacy. It is surprisingly difficult to reason about the behavior of such compositions of components.[5] Attackers may create new side-channels through unexpected connections—for example, a memory DIMM that can send network packets via a shared I2C bus with an Ethernet controller.[17]

Hardware engineers often talk about 'parasitic' resistance or capacitance—components that were not put there by the designer but were created by the physical implementation, often unhelpfully sucking away signals or power. Today we have parasitic computers. Many components have unintended computational power, which can be perverted—from the x86 page-fault handler[2] to DMA controllers.[16] This presents a challenge to understanding where all the computation is happening, such as what is software rather than hardware.

Toward Robustly Engineered Trustworthy Systems

Total-system approaches to security defenses are important (see, for example, Bellovin[3]). A further lesson from physical-layer attacks is why such attacks are not more of a threat today—due to further layers of protection. It is not enough to extract the cryptographic key from a banking card using laser fault injection; the attacker must also use it to steal money. At this point the bank's system-level defenses apply, such as transaction limits and fraud detection. If the key relates only to one account, the payoff involves only money held by that customer, not all other customers. Application-level compartmentalization limits the reward, and thus makes the attack economically nonviable.

Another approach is to ensure that richer contextual information is available that allows the hardware to understand and enforce security properties. The authors are on a team designing, developing, and formally analyzing the CHERI hardware instruction-set architecture,[20] as well as CHERI operating system and application security. The CHERI ISA can enable hardware to enforce pointer provenance, arbitrarily fine-grained access controls to virtual memory and to abstract system objects, as well as both coarse- and fine-grained compartmentalization. Together, these can provide enforceable separation and controlled sharing, allowing trustworthy and untrustworthy software (including unmodified legacy code) to coexist securely. Since the hardware has awareness of software constructs such as pointers and compartments, it can protect them, and we can reason about the protection guarantees—for example, formally proving the architectural abstraction enforces specific security properties. We believe this CHERI system architecture has significant potential to provide unprecedented total-system trustworthiness, including addressing some of the side-channel attacks that were unknown at the time of its conception.[19]

Such architectural guarantees enable more secure implementation of currently insecure languages (such as C/C++) and can put demonstrably secure operating-system kernels on a more secure foundation. Similar approaches may apply in other domains, for example between vulnerable components across a system-on-chip.

Engineering such systems requires a more holistic view, with a tighter interplay between hardware, operating systems and applications. In particular, designers need to understand more of what takes place in layers above or below their field of expertise. Better architectural models enable more robust verification of security properties, and amortizing verification costs across projects helps defenders but not attackers. Such verification must be inclusive, testing all the aspects of a system including the boundaries of implementation-defined behavior.

Better verification can defend us against new vulnerabilities present in the abstractions it is based upon, but not against those that involve phenomena that are not modeled. An open question is whether there is an abstraction between an architectural specification and a full hardware implementation that allows us to fully reason about potential leakage, without being so complex as to being intractable.

Conclusion

Traditional models—in which designers have free reign within tightly constrained layers—are no longer fit for purpose. Hardware/software system security architects need better awareness of what comes above and below them, to be able to reason about what happens at other levels of abstraction, and to understand the effects of composition. Managing overall complexity must fully capture information that might be relevant for security analysis, especially for entirely new classes of vulnerabilities. The defensive battle has only just begun. C

References

1. Armstrong, A. et al. ISA Semantics for ARMv8-A, RISC-V, and CHERI-MIPS. In *Proceedings of the Principles of Programming Languages Conference (POPL)* 2019.
2. Bangert, J. et al. The page-fault weird machine: Lessons in instruction-less computation. In *Proceedings of the USENIX Workshop on Offensive Technologies* (WOOT), 2013.
3. Bellovin, S.M. and Neumann, P.G. The big picture: A systems-oriented view of trustworthiness. *Commun. ACM 61*, 11 (Nov. 2018), 24–26.
4. Beniamini, G. Over The Air: Exploiting Broadcom's Wi-Fi Stack; https://bit.ly/2oA6GJL
5. Gerber, S. et al. Not your parents' physical address space. In *Proceedings of the Hot Topics in Operating Systems Conference (HotOS-XV)* 2015.
6. Goel, S., Hunt, W.A. Jr., and Kaufmann, M. Engineering a formal, executable x86 ISA simulator for software verification. *Provably Correct Systems (ProCoS)*, 2017.
7. Google Project Zero, 2018; https://bit.ly/2CAQzTMGu, R. et al. CertiKOS: An Extensible Architecture for Building Certified Concurrent OS Kernels. OSDI 2016, 653–669; See also https://bit.ly/2Uzj9sI for ongoing work.
8. Islam, S. et al. SPOILER: Speculative Load Hazards Boost Rowhammer and Cache Attacks, arXiv e-prints (Mar. 1, 2019); https://bit.ly/2TxWdhk
9. Klein, G. et al. Comprehensive formal verification of an OS microkernel. *ACM Trans. Computer Systems* 2014; See also https://bit.ly/2UPKgEY for ongoing work.
10. Kocher, P. et al. Spectre attacks: Exploiting speculative execution. ArXiv e-prints (Jan. 2018); https://bit.ly/2lUpJLk
11. Leroy, X. A formally verified compiler back-end. *Journal of Automated Reasoning 43*, 4 (2009), 363–446.
12. Lipp, M. et al. Meltdown, 2018; https://bit.ly/2E6myYl
13. Markettos, A.T. Making sense of the Supermicro motherboard attack; https://bit.ly/2PqOnld
14. Markettos, A.T. et al. Thunderclap: Exploring vulnerabilities in operating system IOMMU protection via DMA from untrustworthy peripherals. In *Proceedings of the Network and Distributed Systems Security Symposium (NDSS)*, (Feb. 2019).
15. Rushanan, M. and Checkoway, S. Run-DMA. In *Proceedings of the WOOT 2015 Conference*. (2015).
16. Sutherland, G. Secrets of the motherboard ([sh*t] my chipset says). In *Proceedings of the 44CON 2017*, (Sept. 2017).
17. Van Bulck, J. et al. Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution. USENIX Security (Aug. 15–17, 2018); https://bit.ly/2DusEDT
18. Watson, R.N.M. et al. Capability Hardware Enhanced RISC Instructions (CHERI): Notes on the Meltdown and Spectre Attacks. Technical Report UCAM-CL-TR-916, University of Cambridge, Computer Laboratory (Feb. 2018); https://bit.ly/2DuVDrr
19. Watson, R.N.M. et al. Capability Hardware Enhanced RISC Instructions (CHERI): CHERI Instruction-set Architecture, Version 7, Technical Report UCAM-CL-TR-927, University of Cambridge, Computer Laboratory (Apr. 2019); https://bit.ly/2XzPgKU
20. Weisse, O. et al. Foreshadow-NG: Breaking the virtual memory abstraction with transient out-of-order execution (Aug. 2018); https://bit.ly/2VZLD0h

A. Theodore Markettos (theo.markettos@cl.cam.ac.uk) is a Senior Research Associate in the Department of Computer Science and Technology at the University of Cambridge, U.K.

Robert N.M. Watson (robert.watson@cl.cam.ac.uk) is a Senior Lecturer in the Department of Computer Science and Technology at the University of Cambridge, U.K.

Simon W. Moore (simon.moore@cl.cam.ac.uk) is Professor of Computer Engineering in the Department of Computer Science and Technology at the University of Cambridge, U.K.

Peter Sewell (Peter.Sewell@cl.cam.ac.uk) is Professor of Computer Science in the Department of Computer Science and Technology at the University of Cambridge, U.K.

Peter G. Neumann (neumann@csl.sri.com) is Chief Scientist of the SRI International Computer Science Lab, and moderator of the ACM Risks Forum.

Calendar of Events

June 2–6
JCDL '19: The 18th ACM/IEEE Joint Conference on Digital Libraries Champaign, IL,
Sponsored: ACM/SIG,
Contact: J. Stephen Downie,
Email: jdownie@illinois.edu

June 3–5
SIGSIM-PADS '19:
SIGSIM Principles of Advanced Discrete Simulation,
Chicago, IL,
Sponsored: ACM/SIG,
Contact: Dong Jin,
Email: dong.jin@iit.edu

June 3–5
SYSTOR '19: International Systems and Storage Conference,
Haifa, Israel,
Sponsored: ACM/SIG,
Contact: Moshik Hershcovitch,
Email: moshikh@il.ibm.com

June 3–6
SACMAT '19: The 24th ACM Symposium on Access Control Models and Technologies,
Toronto, ON,
Sponsored: ACM/SIG,
Contact: Atefeh (Atty) Mashatan,
Email: amashatan@ryerson.ca

June 5–7
TVX '19: ACM International Conference on Interactive Experiences for TV and Online Video,
Salford (Manchester), U.K.,
Sponsored: ACM/SIG,
Contact: Jonathan Hook,
Email: jonathan.hook@york.ac.uk

June 9–12
UMAP '19: 27th Conference on User Modeling, Adaptation and Personalization,
Larnaca, Cyprus,
Co-Sponsored: ACM/SIG,
Contact: George Angelos Papadopoulos,
Email: george@cs.ucy.ac.cy

June 10–13
ICMR '19: International Conference on Multimedia Retrieval,
Ottawa, ON,
Sponsored: ACM/SIG,
Contact: Zhongfei (Mark) Zhang,
Email: zhongfei@cs.binghamton.edu

DOI:10.1145/3325287

Peter J. Denning

The Profession of IT
An Interview with David Brin on Resiliency

Many risks of catastrophic failures of critical infrastructures can be significantly reduced by relatively simple measures to increase resiliency.

Many people today are concerned about critical infrastructures such as the electrical network, water supplies, telephones, transportation, and the Internet. These nerve and bloodlines for society depend on reliable computing, communications, and electrical supply. What would happen if a massive cyber attack or an electromagnetic pulse, or other failure mode took down the electric grid in a way that requires many months or even years for repair? What about a natural disaster such as hurricane, wildfire, or earthquake that disabled all cellphone communications for an extended period?

David Brin, physicist and author, has been worrying about these issues for a long time and consults regularly with companies and federal agencies. He says there are many relatively straightforward measures that might greatly increase our resiliency—our ability to bounce back from disaster. I spoke with him about this.

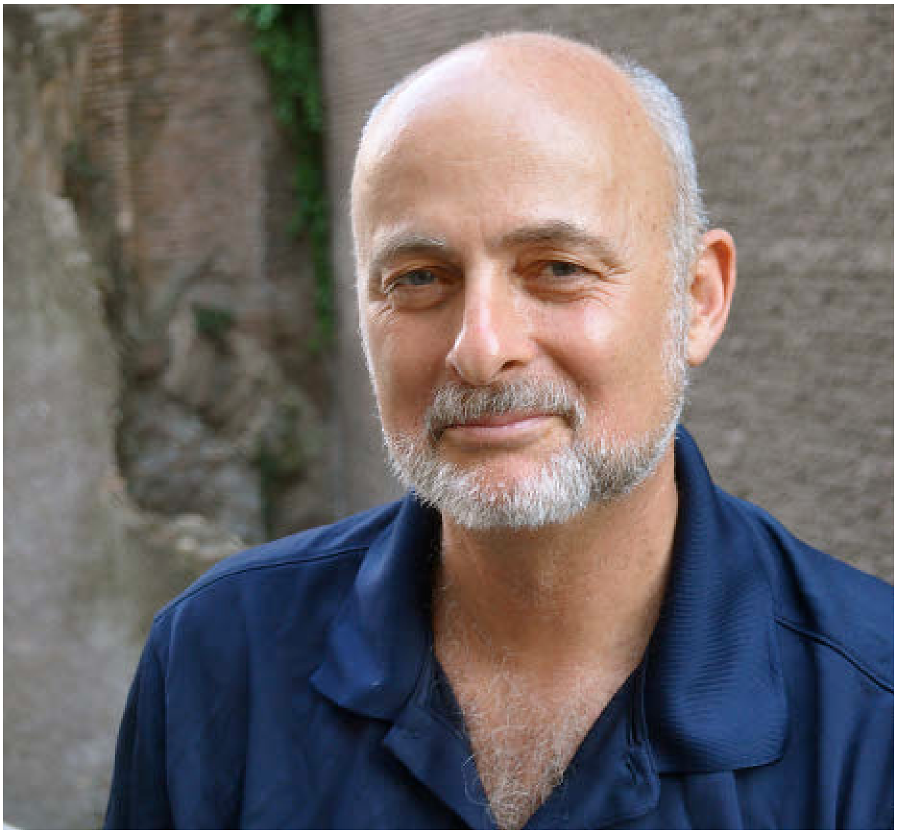


PHOTO BY CHERYL BRIGHAM

Q: What is the difference between resilience and anticipation?

BRIN: Our prefrontal lobes help us envision possible futures, anticipating threats and opportunities. Planners and responders augment these organs with predictive models, intel-gathering, and big data, all in search of dangers to anticipate and counter in advance. Citizens know little about how many bad things these protectors have averted. But this specialization in anticipation makes it hard for protectors to appreciate how we cope when our best-laid plans fail, which they do, sooner or later.

Resilience is how we cope with unexpected contingencies. It enables us to roll with any blow and come up fighting, keeping a surprise from being lethal. It's what worked on 9/11, when all anticipatory protective measures failed.

Q: Let's see what anticipation and resilience look like for a common threat, disruptive electrical outages. They can be caused by storms, birds, squirrels, power grid overload, or even preventive reduction of wildfire risk. Without power, we cannot use our computers or access our files stored in the Internet. Even our best disaster planning cannot fix the disruption if infrastructure damage is severe. Yet, communication is essential

for recovery. What can we do to preserve our ability to communicate?

On 9/11, passengers aboard flight UA93 demonstrated remarkable resilience when they self-organized to stop the terrorist plot to use that plane as a weapon against their country. If we want that kind of resilience to work on a large scale, we need resilient communications. Alas, our comm systems are fragile to failure in any natural or unnatural calamity. One step toward resilience would be a backup peer-to-peer (P2P) text-passing capability for when phones can't link to a cellular tower. Texts would get passed from phone to phone via well-understood methods of packet switching until they encounter a working node and get dropped into the network. Qualcomm already has this capability built into their chips! But cellular providers refuse to turn it on. That's shortsighted, since it would be good business too, expanding text coverage zones and opening new revenue streams. Even in the worst national disaster, we'd have a 1940s-level telegraphy system all across the nation, and pretty much around the world.

All it would take to fix this is a small change of regulation. Five sentences requiring the cell-cos to turn this on whenever a phone doesn't sense a tower. (And charge a small fee for P2P texts.) Doing so might let us restore communications within an hour rather than months.

Many efforts have been made to empower folks with ad hoc mesh networks, via Bluetooth, Wi-Fi webs, and so on. None of these enticed more than a tiny user base—nothing like what's needed for national resilience.

Q: It appears that solar power for homes and offices is at a tipping point as more people find it cheaper than the power grid. Localized solar power should also bring new benefits such as ability to maintain minimum electrical function at home during a blackout. Is independence from the electrical grid good for resilience?

It would be. One can envision a million solar-roofed homes and businesses serving as islands of light for their neighborhoods, in any emergency. But there's a catch. Under current regulations, almost all U.S. solar roofs have a switch that *shuts down* the home or business solar system when the electrical utility has blacked out. The purpose is to prevent spurious home-generated voltages from endangering repair linemen. This is a lame excuse for an insane situation. Simply replace that cutoff switch with one that would still block backflow into the grid, but that feeds from the solar inverter to just two or three outlets inside the home, running the fridge, some rechargers, and possibly satellite coms. Just changing over to that switch would generate archipelagos of autonomous, resilient civilization spread across every neighborhood in America, even in the very worst case. A new rule requiring such switches, and fostering retrofitting, would fit on less than a page.

Across the next decade, more solar systems will come with battery storage. But this reform would help us bridge the next 10 years.

Alas, our comm systems are fragile to failure in any natural or unnatural calamity.

Q: What about protection against electromagnetic pulse disruption?

Much has been written about danger from EMP—either attacks by hostile powers or else the sort of natural disaster we might experience if the Sun ever struck us head-on with a coronal mass ejection, commonly called a solar flare. These CMEs happen often, peaking every 11 years. We've been lucky as the worst ones have missed Earth. But some space probes have been taken out by direct hits and a bulls-eye is inevitable.

The EMP threat was recognized over 30 years ago. We could have incentivized gradual development of shielded and breakered chipsets, including those in civilian electronics. Adoption could have been stimulated with a tax of a penny per non-compliant device, with foreseen ramp-up. By now we'd be EMP resilient, instead of fragile hostages either to enemies or to fate.

Q: What about solar on the southward walls of buildings to power the buildings? Some cities are already doing this.

Sure, south-facing walls are another place for photovoltaics. But there's competition for that valuable real estate—*urban agriculture*. Technologies are cresting toward where future cities may require new buildings to recycle their organic waste through vertical farms that purify water while generating either industrial algae or else much of the food needed by a metropolis. With so much of the world's population going urban, no technology could make a bigger difference. The pieces are coming together. What's lacking is a sense of urgency. Pilot programs and tax incentives should encourage new tall buildings to utilize their southward faces, nurturing this stabilizing trend during the coming decade.

Q: You've also spoken about apps systems that turn your smartphone into an intelligent sensor. Can you say how this supports resiliency?

Cellphones already have powerful cameras, many with infrared capability. Soon will come spectrum-analysis apps, letting citizens do local spot checks on chemical spills or environmental problems, and feeding the results to governments or NGOs for modeling in real time. The Tricorder X Prize showed how just a few add-on devices can turn a phone into a medical appraisal device, like Dr. McCoy had in "Star Trek." Almost anyone could use such apparatus in the field with little training. Take a few measurements, and a distant system advises you on corrective actions.

Infrared sensors, accelerometers, and chemical sensors could provide a full array of environmental awareness systems by turning citizen cellphones into nodes of an instant awareness network. (I describe this in my novel *Existence*.)

Such a mesh is already of interest to national authorities. But the emphasis has been hierarchical—authorities send public reports down to citizens after gathering and interpreting data flowing upward. The hierarchical mind-set comes naturally when you are an authority with protective duties. But this can blind even sincere public servants to one of our great strengths—

the ability of average citizens to self-organize laterally.

Use your imagination. The greatest long-term advantage of our kind of society is that lateral citizen networks, while occasionally inconvenient to public servants, aren't any kind of macro-threat, but will make civilization perform better. This is in contrast to despotic regimes, for whom such citizen empowerment would be lethal.

Q: Some of your proposals are less familiar. You have spoken of "all sky awareness." What is that and how does it improve resiliency?

Defense and intelligence folks know we need better 24/7 omni-awareness of land, sea, and air. Major efforts involve protective services and space assets. When the Large Synoptic Telescope comes online in Chile, we'll find 100 times as many asteroids that could threaten our planet, or like the one that broke 10,000 windows in Chelyabinsk. Closer to home, dangerous space debris should be tracked round the globe.

Similar technology could improve air safety and impede smugglers by tracking both legal and illicit air traffic. For example, the cell networks I mentioned earlier could detect and triangulate aircraft engine sounds for comparison to an ongoing database, especially at low altitudes where drug smugglers and human traffickers operate, or where terrorists might attempt an attack, or detecting the path of airliners that stray, like Malaysian Air flight 370. Imagine those in peripheries like Canada, Alaska, or nearby waters automatically reporting sonic booms. Among myriad more mundane uses, these might perhaps localize incoming hypersonic weapons, of the kind announced recently by Russian President Vladimir Putin.

Sound implausible? In December 2018, a loose network of amateur 'plane-spotters' managed to track Air Force One visually, during President Trump's top-secret Christmas dash to a U.S. air base in Iraq. A U.K. photographer used these clues to snap the unmistakable, blue-and-white 747 jetting far overhead.

Another method: revive the SETI League's Project Argus, aiming to establish radio and optical detectors in 5,000 amateurs' backyards, spread around the world. As Earth rotates, these backyard stations would sweep the sky in overlapping swathes, sifting for anomalous signals, but also detecting almost anything interesting that happens up there. Argus failed earlier because of the complexity and expense of racks of equipment. Today—with a small up-front investment by some mere-millionaire—we could offer a small box for a couple of hundred bucks that could be latched to an old TV dish-antenna, then Wi-Fi linked via the owner's home. The dish—plus a small optical detector—could report detections in real time and any pair or trio that correlate would then trigger a look by higher-level, aimable devices.

Sure, most of the participants would think of their backyard SETI stations as helping sift the sky for aliens. So? As a side benefit, we'd become hundreds of times better at detecting almost any transient phenomenon overhead, improving both anticipation and resilience.

I can go on with a much longer list of unconventional and generally very inexpensive ways that very simple regulatory or incentive actions might transform national resilience, making society more robust to withstand shocks across the decades ahead.

Q: What about civil unrest or lawlessness if the disaster takes out or overwhelms local law enforcement? Easy to see gangs roaming affluent neighborhoods in SUVs stealing stuff and especially food, with no police to stop them.

Sincere people across the spectrum are right to worry about companies and governments collecting massive amounts of personal data on citizens.

I well-understand this worry! I've written collapse-of-civilization tales. (One of them, *The Postman*, was filmed by Kevin Costner.) Hollywood presents so many apocalyptic scenarios, we tend to assume we live on a fragile edge of collapse. But Rebecca Solnit's book, *A Paradise Built In Hell*, shows decisively that average citizens—whether liberal or conservative—are actually pretty tough and dynamic. They quickly self-organize to help their neighbors. A quarter or more of citizens will almost always run **toward** whatever the problem is. Take citizen response on 9/11, or when disasters hit their neighborhoods.

If "affluent neighborhoods" want to be safe, there's one method that works over the long run … don't alienate the poor and middle class and ensure that the vast majority identify as members of the same overall tribe. As neighbors, we'll come to your defense.

Q: Anything to mitigate cyber attacks, including phishing and massive identity theft?

Sincere people across the spectrum are right to worry about companies and governments collecting massive amounts of personal data on citizens: from the ways they use their smartphones, to always-on mics at home and office (for example, Alexa). Phishing is another example where crooks use already open knowledge about you to lure you into fatal online mistakes. We all fret about disparities of power that may lead to the "telescreen" in George Orwell's *Nineteen Eighty-Four*. From facial recognition to video fakery to brainwave interpretation and lie detectors, if these techs are monopolized by one elite or another, we may get Big Brother forever. There are forces in the world who are eager for this. China's "social credit" system aims to the masses to enforce conformity on one another.

In the West, most people are right to find this prospect terrifying. The reflex in response is to say: "let's ban or restrict this new kind of light." And that is the worst possible prescription. The elites we fear will only gain great power if they can operate in secret, enhancing that disparity, because we won't be able to look back.

Consider. It matters much less what elites of all kinds *know* about you than what they can *do* to you. And the only thing that deters the latter is what *we* know about *them*. Denying elites the power to see has never happened (for long) anywhere in the history of the world. But denying them the ability to harm citizens is something we've (imperfectly) accomplished for 200 years. We've done it by insisting that *we* get to see, too. If not as individuals, then via the NGOs we hire to look for us.

As I appraise in *The Transparent Society*, the answer is *more* light, not less, for common citizens to be empowered by technology to take up much of the burden of supervising and arguing and applying accountability. The more we can see the less the bad groups can hide. If we do this, we'll not only be resilient, we'll *never* have Big Brother.

The answer to phishing, ID theft, etc., is the same as always—to catch and deter villains, by ending most shadows for roaches to hide in.

Q: We don't know how to do this because the Internet itself is baked in a cloak of anonymity. We are not going to redesign the Internet protocols anytime soon. We need more than light. Isn't the solution good locks on our databases?

Sorry, show me one time when "good locks" worked for very long. Every week, some previously "for sure" database is raided or leaks. All that needs happen is for any lock to fail once, at all, via code-breaking or hacking or phishing or human error, and the information is loose, infinitely copyable. If you base your sense of safety on secrecy, it will be impossible to verify what others *don't know*.

Look, I'm not saying that there should be no secrets or privacy! Our skilled protectors need tactical secrecy to do their jobs. But smaller volumes and perimeters are easier to defend and seal. It has always been U.S. policy that secrecy should bear some burden of justification and—eventually—a time limit.

This isn't the time or place to argue the point. Alas, the reflex to seek safety in shadows is so strong that folks forget how we got the very freedoms, wealth, and justice we worry about losing. Not by *hiding* but by assertively demanding to see. What I do ask is that you squint and look ahead 50 or 100, and ask *what is our baseline victory condition*?

In an era of high tech and lightning reaction times, we must rely on a highly professional cadre of protectors.

Every enemy of this enlightenment, individualist, open-society experiment—every lethal foe—is mortally allergic to light. They suffer when their plans, methods, agents, and resources are revealed. In contrast, we are at worst inconvenienced and—as shown by the Snowden and WikiLeaks affairs—even prodded to improve a bit. If, say in 50 years, there is worldwide transparency of ownership and power and action, then we win. We—a humanity that is inquisitive, confident, individualistic, and free—simply win.

Q: These resiliency proposals all sound so reasonable. Why have they not been implemented?

A cynic would answer that there's not much economic-constituency behind resilience. No big-ticket orders. How much money is to be made from a slightly costlier home-solar cutoff switch that would feed rooftop energy to three outlets in a million U.S. homes? I spoke about backup peer-to-peer texting at a defense industry conference where a Verizon vice-president in attendance went absolutely livid. Qualcomm tried subsequently to get them—and AT&T—to try some regional experiments; might P2P texting might actually turn a profit? Alas, no one wants to risk disruption, even though this one function could knit our entire continent together, in a crisis.

EMP resistance should have been slowly woven into civilian electronics for decades. And here's a thought—maybe it has been! After all, if we had truly savvy leaders, they would want to slide this protection into place as quietly as possible. Why? Because there is a critical vulnerability window, during which those who are thinking about hitting us might strike if they see the chance slipping away. History shows that such transitions can be dangerous, as revealed by John F. Kennedy in *While England Slept*.

Some bright folks are paying attention. Elon Musk told me he would fix the solar cutoff problem with his Power Wall storage system, and that *is* the answer ... in a decade. A $200 switch would still be worthwhile, till then. Another zillionaire expressed interest in the all-sky awareness project, but more for its contribution to SETI than national or world security. Membership in CERT—Community Emergency Response Teams—rises every year. And so it goes. Just way too slowly.

What truly matters is the very concept of resilience, which worked so well on 9/11 and at every turn of American history. The U.S. Army, till just one generation ago, always based its planning on vast pools of talented, healthy volunteers rushing in to fill the thin blue line. Sure, in an era of high tech and lightning reaction times, we must rely on a highly professional cadre of protectors. But the worst thing they could do is to declare "Count on us ... and *only* on us."

No. We love you and thank you for your service. But a time will come when you will fail. And when that happens, it will be our turn—*citizens*—to step up.

Help us to prepare, and we won't let you down. C

David Brin (http://www.davidbrin.com) is an astrophysicist whose international best-selling novels include *The Postman*, *Earth*, and *Existence*. He serves on advisory boards (for example, NASA's Innovative and Advanced Concepts program or NIAC) and speaks or consults on a wide range of topics including AI, SETI, privacy, and national security. His nonfiction book about the information age—*The Transparent Society*—won the Freedom of Speech Award of the American Library Association.

Peter J. Denning (pjd@nps.edu) is Distinguished Professor of Computer Science and Director of the Cebrowski Institute for information innovation at the Naval Postgraduate School in Monterey, CA, is Editor of ACM *Ubiquity*, and is a past president of ACM. The author's views expressed here are not necessarily those of his employer or the U.S. federal government.

DOI:10.1145/3322933

Thomas Pasquier, David Eyers, and Jean Bacon

Viewpoint
Personal Data and the Internet of Things

It is time to care about digital provenance.

We have all read market predictions describing billions of devices and the hundreds of billions dollars in profit that the Internet of Things (IoT) promises.[a] Security and the challenges it represents[27] are often highlighted as major issues for IoT, alongside scalability and standardization. In 2017, FBI Director James Comey warned, during a senate hearing, of the threat represented by a botnet taking control of devices owned by unsuspecting users. Such a botnet can seize control of devices ranging from connected dishwashers,[b] to smart home cameras and connected toys, not only using them as a platform to launch cyber-attacks, but also potentially harvesting the data such devices collect.

In addition to concerns about cybersecurity, corporate usage of personal data has seen increased public scrutiny. A recent focus of concern has been connected home hubs (such as Amazon Alexa and Google Home).[c] Articles on the topic discussed whether conversations were being constantly recorded and if so, where those records went. Similarly, the University of Rennes faced a public backlash after revealing its plan to deploy smart-beds in its accommodation to detect "abnormal" usage patterns.[d] A clear question emerges from IoT-related fears: "How and why is my data being used?"

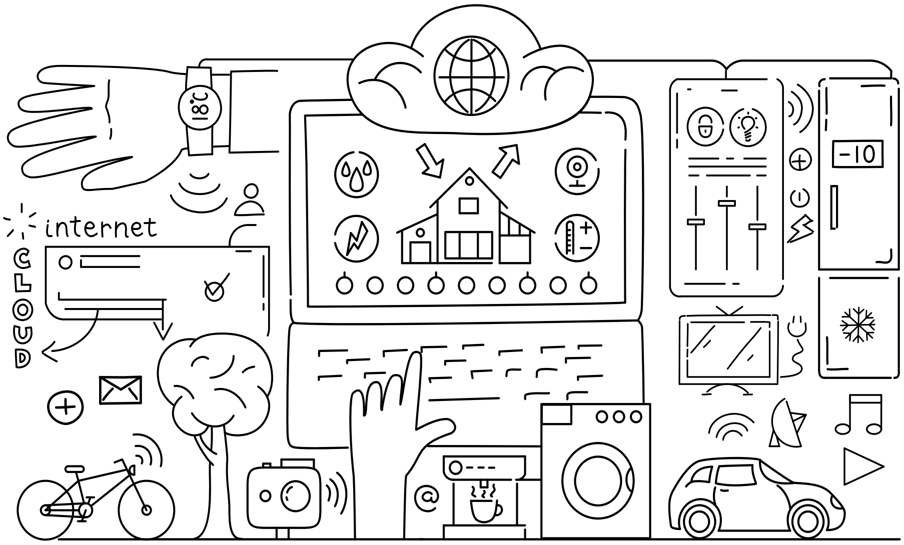


IMAGE BY KOSTENKO MAXIM

As concerns grow, legislators across the world are taking action in order to protect the public. For example, the recent EU General Data Protection Regulation (GDPR) that took effect in May 2018,[e] and the forthcoming ePrivacy Regulation[f] place strong responsibility on data controllers to protect personal data, and to notify users of security breaches. The EU commission defines a Data Controller as the party that determines the purposes for which, and the means by which, personal data is processed (*why* and *how* the data is processed). EU regulations further impose constraints on EU citizens' data processing based on location and data type (that is, "special category" data falls under more stringent constraints). The data controller must provide means for end users to determine whether their data is properly handled and means to effect their rights. Overall, there must be mechanisms to determine what data is processed, *how*, *why*, and *where*.

Such concerns have drawn researchers to look at means to develop more accountable and transparent systems.[10,24] The problem has also been clearly highlighted by the EU Data Protection Working Party: "As a result of the need to provide pervasive services in an unobtrusive manner, users might in practice find themselves under third-party monitoring. This may result in situations where the user can lose all control on the dissemination of his/her data, depending on whether or not the collection and processing of this data will be made in a transparent manner or not."

[a] See https://bit.ly/2JNx0LZ
[b] See https://bit.ly/2JIOidc
[c] See https://bit.ly/2gY9qKG
[d] See https://lemde.fr/2HLvEQb

[e] See https://bit.ly/2lSJQfO
[f] See https://bit.ly/2j4AwzT

Modern computing systems contain many components that operate as black boxes; they accept inputs and generate outputs but do not disclose their internal working. Beyond privacy concerns, this also limits the ability to detect cyber-attacks, or more generally to understand cyber-behavior. Because of these concerns DARPA, in the U.S., launched the Transparent Computing project[g] to explore means to build more transparent systems through the use of digital provenance with the particular aim of identifying advanced persistent threats. While DARPA's work is a good start, we believe there is an urgent need to reach much further. In the remainder of this Viewpoint, we explore how provenance can be an answer to some IoT concerns and the challenges faced to deploy provenance techniques.

Digital Provenance

There is a growing clamor for more transparency, but straightforward, widespread technical solutions have yet to emerge. Typical software log records often prove insufficient to audit complex distributed systems as they fail to capture the complex causality relationships between events. Digital provenance[8] is an alternative means to record system events. Digital provenance is the record of information flow within a computer system in order to assess the origin of data (for example, its quality or its validity).

The concept first emerged in the database research community as a means to explain the response to a given query.[16] Provenance research later expanded to address issues of scientific reproducibility, notably by providing mechanisms to reconstitute computational environments from formal records of scientific computations.[23] More recently, provenance has been explored within the cybersecurity community[25] as a means to explain intrusions[18] or more recently to detect them.[14]

Provenance records are represented as a directed acyclic graph that shows causality relationships between the states of the objects that compose a complex system. As a consequence, it is compatible with automated mathematical reasoning. In such a graph, the vertices represent the state of transient and persistent data items, transformations applied to those states, and persons (legal or natural) responsible for data and transformations (generally referred to as entities, activities, and agents respectively). The edges represent dependencies between these entities. The analysis of such a graph allows us to understand *where*, *when*, *how*, *by whom*, and *why* data has been used.[7,9]

An outcome of research on provenance in the cybersecurity space is the understanding that the capture mechanism must provide guarantees of completeness (all events in the system can be seen), accuracy (the record is faithful to events) and a well-defined, trusted computing base (the threat model is clearly expressed).[22] Otherwise, attacks on the system may be undetected, dissimulated by the attacker, or misattributed. We argue that in a highly ad hoc and interoperable environment with mutually untrusted parties, the provenance used to empower end users with control and understanding over data usage requires similar properties.

Who to Trust?

In the IoT environment the number of involved stakeholders has the potential to explode exponentially. Traditionally, a company managed its own server infrastructure, maybe with the help of a subcontractor. The cloud computing paradigm further increased complexity with the involvement of cloud service providers (sometimes stacked, for example, Heroku PaaS on top of the Amazon IaaS cloud service), third-party service providers (for example, CloudMQTT) and other tenants sharing the infrastructure. The IoT further increases this complexity, with potentially ad hoc and unforeseen interactions between devices and services on top of the complex cloud and edge computing infrastructure most IoT services rely on.

One answer to this problem is to build applications in "silos" where the involved parties are known in advance, but as a side-effect locking-in devices and services to a single company (for example, the competing smart-home offerings by leading technology companies). This is far from the IoT vision of a connected environment, but most existing products fall into this category. There are obviously major business considerations behind this model, and it should be noted that the EU GDPR mandates for some form of interoperability (although it is yet unclear how it should be interpreted[12]).

An alternative to such "lock-in" would be to make devices' consumption of data transparent and accountable. If data is exchanged across devices, the concerned user should be able to audit its usage. However, in an environment where arbitrary devices could interact (although it must be remembered that EU GDPR requires explicit and informed user consent), how can trust be established in the audit record? This requires an in-depth rethinking of how IoT platforms are designed, potentially exploring the security-by-design approach based on hardware roots of trust[13] to provide trusted digital enclaves in which behavior can be audited. Some form of "accountability-by-design" principle should also be encouraged, where transparency and the implementation of a trustworthy audit mechanism is a core concern in product design.

Such solutions have been explored in the provenance space, for example, by leveraging SGX properties to provide a strong guarantee of the integrity of the provenance record.[4] Similarly, remote attestation techniques leveraging TPM hardware have been proposed[6] to guarantee the integrity of the capture mechanism. However, how to provide such guarantees in an IoT environment, where such hardware features may not be available, is a relatively unexplored topic.

Where Does the Audit Live?

The fully realized IoT vision is of vast distributed and decentralized systems.

Building transparent and auditable systems may be one of the greatest software engineering challenges of the coming decade.

[g] See https://bit.ly/2Uf5bQY

If we assume trustworthy provenance capture is achievable, the issue of guaranteeing that the provenance record can be audited remains. If you are to audit the processing of personal data, guarantees about the integrity and availability of the provenance record must exist. If you agreed to share your daily activity for research, the activities of insurance companies scraping your data for possible health risks must not be able to masquerade as benign research use, nor should data collection for political purposes be able to pass as harmless entertainment, as in the Cambridge Analytica scandal.[h] Similarly, the availability (durability) of the audit record must be guaranteed. There is no point to an audit record if it can simply be deleted.

Further, Moyer et al. evaluated the storage requirements of provenance when used for security purposes in relatively modest distributed systems.[21] In such a context, several thousands of graph elements can be generated per second and per machine, resulting in a graph containing billions of nodes to represent system execution over several months. It is unclear how some past research outcomes, for example, detection of suspicious behavior,[2] privacy-aware provenance[11] or provenance integrity,[15] scale to very large graphs, as such concerns were not evaluated. Similarly, while blockchain is heralded[19] as an integrity-preserving means to store provenance, it is unclear how well it could expand to such scale. Several options have been explored to reduce graph size, such as identifying and tracking only sensitive data objects[5] or performing property-preserving graph compression[17] however none has yet adequately addressed the scalability challenge.

How to Communicate Information?

Means must be developed to communicate about data usage, but also about the risks of inference from the data. Not only must the nature of the data be considered, but also other properties such as the frequency of capture.[3] For example, a 100Hz smart-meter reading can in some cases indicate what television channel is currently being watched; even a daily average reading could inform about occupancy. Here, it is important to be able to explore and represent the outcome of complex computational workflow.[1]

Provenance visualization has been an active research topic for over a decade, yet no fully satisfactory solution has been proposed. The simplest possible visualization is to render the graph, however beyond trivially simple graphs such a representation is too complex and dense to be easily understood, even by experts. We go further and suggest that how interpretable such information is for end users also depends on educational background, socioeconomic environment, and culture.

In order to make the accountability and transparency of IoT platforms effective, a better communication medium must be provided. An approach often taken is to analyze motifs in the graph to extract high-level abstractions (for example, Missier et al.[20]), meaningful to the average end user. In recent work, it was proposed to represent such a high-level abstraction as a comic strip.[26]

We Need to Care About Digital Provenance

Building transparent and auditable systems may be one of the greatest software engineering challenges of the coming decade. As a consequence, digital provenance and its application to cybersecurity and the management of personal data has become a hot research topic. We have highlighted key active areas of research and their associated challenges. It is fundamental for industry practitioners to understand the threat posed by the black-box nature of the IoT, the potential solutions, and the challenges to a practical deployment of those solutions. Accountability-by-design must become a core objective of IoT platforms.

[h] See https://nyti.ms/2HH74vA

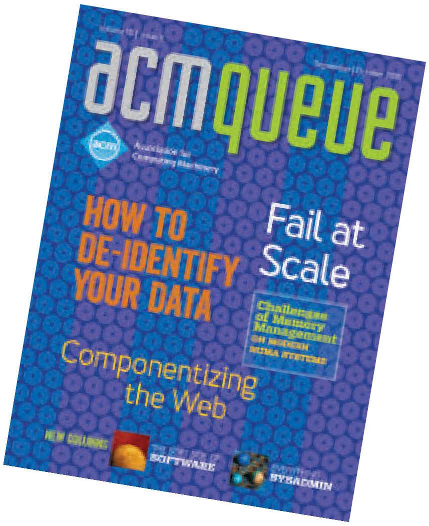
References

1. Acar, U. et al. A graph model of data and workflow provenance. In *Proceedings of the TAPP'10 Second Conference on Theory and Practice of Provenance*, USENIX, 2010.
2. Allen, M.D. et al. Provenance for collaboration: Detecting suspicious behaviors and assessing trust in information. In *Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*. IEEE, 2011, 342–351.
3. Amar, Y. et al. An information theoretic approach to time-series data privacy. In *Proceedings of the 1st Workshop on Privacy by Design in Distributed Systems*. ACM, (2018), 3.
4. Balakrishnan, N. et al. Non-repudiable disk I/O in untrusted kernels. In *Proceedings of the 8th Asia-Pacific Workshop on Systems*. ACM, 2017, 24.
5. Bates, A. et al. Take only what you need: Leveraging mandatory access control policy to reduce provenance storage costs. In *Proceedings of the Conference on Theory and Practice of Provenance* (2015), USENIX, 7–7.
6. Bates, A.M. et al. Trustworthy whole-system provenance for the Linux kernel. In *Proceedings of the USENIX Security Symposium* (2015) 319–334.
7. Buneman, P. et al. Why and where: A characterization of data provenance. In *Proceedings of the International Conference on Database Theory*. Springer, 2001, 316–330.
8. Carata, L. et al. A primer on provenance. *Commun. ACM 57*, 5 (May 2014), 52–60.
9. Cheney, J. et al. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases 1*, 4 (2009), 379–474.
10. Crabtree, A. et al. Building accountability into the Internet of Things: The IoT databox model. *Journal of Reliable Intelligent Environments* (2018).
11. Davidson, S. et al. Provenance views for module privacy. In *Proceedings of the Thirtieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. ACM, 2011, 175–186.
12. De Hert, P. et al. The right to data portability in the GDPR: Towards user-centric interoperability of digital services. *Computer Law & Security Review*. Elsevier, (2017).
13. Eldefrawy, K. et al. SMART: Secure and minimal architecture for (establishing dynamic) root of trust. In *Network and Distributed System Security Symposium 12* (2012), 1–15.
14. Han, X. et al. FRAPpuccino: Fault-detection through Runtime Analysis of Provenance. In *Proceedings of the Workshop on Hot Topics in Cloud Computing (HotCloud'17)*. USENIX (2017).
15. Hasan, R. et al. The case of the fake Picasso: Preventing history forgery with secure provenance. In *Proceedings of the Conference on File and Storage Technologies (FAST'09)*, (2009), 1–14.
16. Herschel, M. et al. A survey on provenance: What for? What form? What from? *The VLDB Journal—The International Journal on Very Large Data Bases 26*, 6 (2017), 881–906.
17. Hossain, M.N. et al. Dependence-preserving data compaction for scalable forensic analysis. In *Proceedings of the USENIX Security Symposium*.
18. King, S.T. and Chen, P.M. Backtracking intrusions. *ACM SIGOPS Operating Systems Review 37*, 5 (May 2003).
19. Liang, X. et al. Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In *International Symposium on Cluster, Cloud and Grid Computing*. IEEE/ACM, (2017), 468–477.
20. Missier, P. et al. ProvAbs: Model, policy, and tooling for abstracting PROV graphs. In *Proceedings of the International Provenance and Annotation Workshop*. Springer, 2017, 3–15.
21. Moyer, T. and Gadepally, V. High-throughput ingest of data provenance records into Accumulo. In *Proceedings of the High Performance Extreme Computing Conference (HPEC)*, IEEE, 2016, 1–6.
22. Pasquier, T. et al. Runtime analysis of whole system provenance. In *Proceedings of the Conference on Computer and Communications Security (CCS'18)*. ACM, 2018.
23. Pasquier, T. et al. If these data could talk. *Scientific Data 4* (2017), http://www.nature.com/sdata2017114.
24. Pasquier, T. et al. Data provenance to audit compliance with privacy policy in the Internet of Things. *Personal and Ubiquitous Computing* (2018), 333–344.
25. Pohly, D.J. et al. Hi-Fi: Collecting high-fidelity whole-system provenance. In *Proceedings of the 28th Annual Computer Security Applications Conference*. ACM, 2012, 259–268.
26. Schreiber, A. and Struminski, R. Tracing personal data using comics. In *Proceedings of the International Conference on Universal Access in Human-Computer Interaction*. Springer, 2017, 444–455.
27. Singh, J. et al. Twenty security considerations for cloud-supported Internet of Things. *IEEE Internet of Things Journal 3*, 3 (Mar. 2016), 269–284.

Thomas Pasquier (http://tfjmp.org) is a Lecturer (Assistant Professor) at the University of Bristol's Cyber Security Group, and a visiting scholar at the University of Cambridge, U.K.

David Eyers (https://www.cs.otago.ac.nz/staff/David_Eyers) is an Associate Professor in the Department of Computer Science at the University of Otago, New Zealand.

Jean Bacon (http://www.cl.cam.ac.uk/~jmb25/) is Professor Emerita of Distributed Systems at the University of Cambridge, U.K.


acmqueue
HOW TO DE-IDENTIFY YOUR DATA
Fail at Scale
Challenges of Memory Management on Modern NUMA Systems
Componentizing the Web
SOFTWARE
SYSADMIN

DOI:10.1145/3316772

Article development led by acmqueue
queue.acm.org

A collaborative approach to reclaiming memory in heterogeneous software systems.

BY ULAN DEGENBAEV, MICHAEL LIPPAUTZ, AND HANNES PAYER

Garbage Collection as a Joint Venture

MANY POPULAR PROGRAMMING languages are executed on top of virtual machines (VMs) that provide critical infrastructure such as automated memory management using garbage collection. Examples include dynamically typed programming languages such as JavaScript and Python, as well as static ones like Java and C#. For such languages the garbage collector periodically traces through objects on the application heap to determine which objects are live and should be kept or dead and can be reclaimed.

The garbage collector is said to manage the application memory, which means the programming language is managed. The main advantage of managed languages is that developers do not have to reason about object lifetimes and free objects manually. Forgetting to free objects leaks memory, and premature freeing results in dangling pointers.

Virtual machines for managed languages may be embedded into larger software systems that are implemented in a different, sometimes unmanaged, programming language, where programmers are responsible for releasing memory that is no longer needed. An example of such a heterogenous software system is Google's Chrome Web browser where the high-performance V8 JavaScript VM (https://v8.dev/) is embedded in the Blink rendering engine that is in charge of rendering a website. Blink renders these pages by interpreting the document object model (DOM; https://www.w3.org/TR/WD-DOM/introduction.html) of a website, which is a cross-platform language-independent representation of the tree structure defined through HTML.

IMAGE BY WACOMKA

Since Blink is written in C++, it implements an abstract DOM representing HTML documents as C++ objects. The DOM C++ objects are wrapped and exposed as objects to JavaScript, which allows scripts to manipulate Web page content directly by modifying the DOM objects. The C++ objects are called *wrappables*, their JavaScript counterparts *wrappers*, and the references connecting these objects *cross-component references*. Even though C++ is an unmanaged language, Blink has its own garbage collector for DOM C++ objects. Cross-component memory management then deals with reclaiming memory in such heterogeneous environments.

V8 and Blink use mark-sweep-compact garbage collectors where a single garbage-collection cycle consists of three phases: *marking*, where live objects are identified; *sweeping*, where dead objects are released; and *compaction*, where live objects are relocated to reduce memory fragmentation. During marking, the garbage collector finds all objects reachable from a defined set of root references, conceptually traversing an object graph, where the nodes of the graph are objects and the edges are fields of objects.

Cross-component references express liveness over component boundaries and have to be modeled explicitly in the graph. The simplest way to manage those references is by treating them as roots into the corresponding component. In other words, references from Blink to V8 would be treated as roots in V8 and vice versa. This creates the problem of reference cycles across components, which is analogous to regular reference cycles[1] within a single garbage-collection system, where objects form groups of strongly connected components that are otherwise unreachable from the live object graph.

Cycles require either manual breaking through the use of weak references or the use of some managed system able to infer liveness by inspecting the system as a whole. Manually breaking a cycle is not always an option because the semantics of the involved objects may require all their referents to stay alive through strong references. Another option would be to restrict the involved components in such a way that cycles cannot be constructed. Note that in the case of Chrome and the Web this is not always possible, as shown later.

While the cycle problem can be avoided by unifying the memory-management systems of two components, it may still be desirable to manage the memory of the two components independently to preserve separation of concerns, since it is simpler to reuse a component in another system if there are fewer dependencies. For example, V8 is used not only in Chrome, but also in the Node.js server-side runtime, making it undesirable to add Blink-specific knowledge to V8.

Assuming the components cannot be unified, the cross-component reference cycles can lead to either *memory leaks* when graphs involving cycles cannot be reclaimed by the components' garbage collectors, heavily impacting browser performance, or *premature collection of objects* resulting in use-after-free security vulnerabilities and program crashes that put users at risk.

This article describes an approach called cross-component tracing (CCT),[3] which is implemented in V8 and Blink to solve the problem of memory management across component boundaries. Cross-component tracing also integrates nicely with existing tooling infrastructure and improves the debugging capabilities of Chrome DevTools (https://developers.google.com/web/tools/chrome-devtools/).

Cross-component tracing enables efficient, effective, and safe garbage collection across component boundaries.

Separate Worlds for DOM and JavaScript

As mentioned, Chrome encodes the DOM in C++ wrappable objects, and most functionality specified in the HTML standard is provided as C++ code. In contrast, JavaScript is implemented within V8 using a custom object model that is incompatible with C++. When JavaScript application code accesses properties of JavaScript DOM wrapper objects, V8 invokes C++ callbacks in Blink, which make changes to the underlying C++ DOM objects. Conversely, Blink objects can also directly reference JavaScript objects and modify those as needed. For example, Blink can bind fields of JavaScript objects to C++ callbacks that can be used by other JavaScript code.

Both worlds—DOM and JavaScript—are managed by their own trace-based garbage collectors able to reclaim memory that is only transitively rooted within their own heaps. What remains is defining how cross-component references should be treated by these garbage collectors to enable them to effectively collect garbage across components To highlight the problems of leaks and dangling pointers, it is useful to look at a concrete example of JavaScript code and how it can be used to create dynamic content that changes over time.

Figure 1 shows an example that creates a temporary object, a loading bar (`loadingBar`), that is then replaced by actual content (`content`) asynchronously built and swapped in as soon as it is ready. Note that accessing the document element or the body element, or creating the div elements results in pairs of objects in their respective worlds that hold references to each other. While the program itself is written in JavaScript, property look-ups to, for example, the body element and calls to DOM methods `appendChild` and `replaceChild` are forwarded to their corresponding C++ implementations in Blink. Regular JavaScript access, such as setting a parent property, is carried out by V8 on its own objects. It is this seamless integration of JavaScript and the DOM that allows developers to create rich Web applications. At the same time, this concept allows the creation of arbitrary object graphs across component boundaries.

Figure 2 shows a simplified version of the object graph created by the example, where JavaScript objects on the left are connected to their C++ counterparts in the DOM on the right. Java-Script objects, such as the body and div elements, have hardly any references in JavaScript but are mostly used to refer to their corresponding C++ objects. It is thus crucial to define the semantics of cross-component references for the component-local garbage collectors to allow collection of these objects. For example, treating incoming references from Blink into V8 as roots for the V8 garbage collector would always keep the `loadingBar` object alive. Treating such references as uniformly weak would result in reclamation of the body and the div elements by the V8 garbage collector, which would leave behind dangling pointers for Blink.

Besides correctness, another challenge in such an entangled environment is debuggability for developers. While the Web platform allows loose coupling of C++ and JavaScript under the hood, it is crucial that the APIs for these abstractions are properly encapsulated for Web developers who use HTML and JavaScript, including preventing memory leaks when properly used. To investigate memory leaks in Web pages, developers need tools that allow them to reason seamlessly about the connectivity of objects spanning both V8 and Blink heaps.

Cross-Component Tracing

We propose CCT as a way to tackle the general problem of reference cycles across component boundaries. For CCT, the garbage collectors of all involved components are extended to allow tracing into a different component, managing objects of potentially different programming languages. CCT uses the garbage collector of one component as the *master tracer* to compute the full transitive closure of live objects to break cycles.

Other components assist by providing a *remote tracer* that can traverse the objects of the component when requested by the master tracer. The system can then be treated as one managed heap. As a consequence, the simple algorithm of CCT can be extended to allow moving collectors and incremental or concurrent marking as needed by just following existing garbage collection principles.[8] The pseudocode of the master and remote tracer algorithms is available in our full research article.[3]

For Chrome we developed a version of cross-component tracing where the master tracer for JavaScript objects and the remote tracer for C++ objects are provided by V8 and Blink, respectively. This way V8 can trace through the C++ DOM upon doing a garbage collection, effectively breaking cycles on the V8 and Blink boundary. In this system, Blink garbage collections deal with only the C++ objects and treat the incoming cross-component references from V8 as roots. This way, subsequent invocations of V8's and Blink's garbage collectors can reclaim cycles across the component boundary.

The tracer in V8 makes use of the concept of hidden classes[2] that describe the body of JavaScript objects to find references to other objects, as well as to Blink. The tracer in Blink requires each garbage-collected C++ class to be manually annotated with a method that describes the body of the class, including any references to other managed objects. Since Blink was already garbage-collected before introducing CCT, only minor adjustments to this method were required across the rendering codebase.

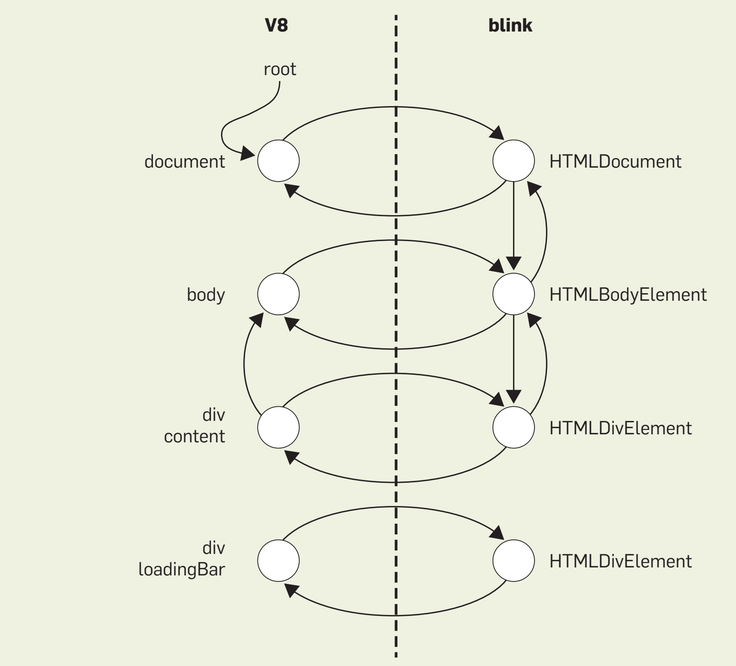
Chrome strives to provide smooth user experiences, updating the screen at 60fps (frames per second), leaving V8 and Blink around 16.6 milliseconds to render a frame. Since marking large heaps may take hundreds of milliseconds, both V8 and Blink employ a technique called *incremental marking*, which means that marking is divided into steps during which objects are marked for only a small amount of time (for example, 1ms).

The application is free to change object references between the steps. This means that the application may hide a reference to an unmarked object in an already-marked object, which would result in premature collection of a live object. Incremental marking requires a garbage collector to keep the marking state consistent by preserving the strong tri-color-marking invariant.[8] This invariant states that fully marked objects are allowed to point only to

Figure 1. JavaScript example interacting with the DOM.

```
<!DOCTYPE html>
<html>
  <body><script>
    function fetchContent(callback) {
      // Emulate network request and content creation.
      setTimeout(callback, 1000);
    }
    function run() {
      const loadingBar = document.createElement(“div”);
      document.body.appendChild(loadingBar);
      fetchContent(() => {
        const content = document.createElement(“div”);
        document.body.replaceChild(content, loadingBar);
        content.parent = document.body;
      });
    }
    document.addEventListener(“DOMContentLoaded”, run);
  </script></body>
</html>
```

Figure 2. Object graph spanning JavaScript and the DOM.



objects that are also fully marked or stashed somewhere for processing. V8 and Blink preserve the marking invariant using a conservative Dijkstra-style write barrier[6] that ensures that writing a value into an object also marks the value. In fact, V8 even provides concurrent marking on a background thread this way while relying on incremental tracing in Blink.[5]

To make this concrete, Figure 3 illustrates CCT where V8 traces and marks objects in JavaScript, as well as C++. Objects transitively reachable by V8's root object are marked black. Subsequently, any unreachable objects (`loadingBar`, in this example) are reclaimed by the garbage collector. Note that from V8's point of view, there is no d ifference between the div elements `content` and `loadingBar`, and only CCT makes it clear which object can be reclaimed by V8's garbage collector. Once the unreachable V8 object is gone, any subsequent garbage collections in Blink will not see a root for the corresponding `HTMLDivElement` and reclaim the other half of the wrapper-wrappable pair.

Figure 3. Cross-component garbage collection.

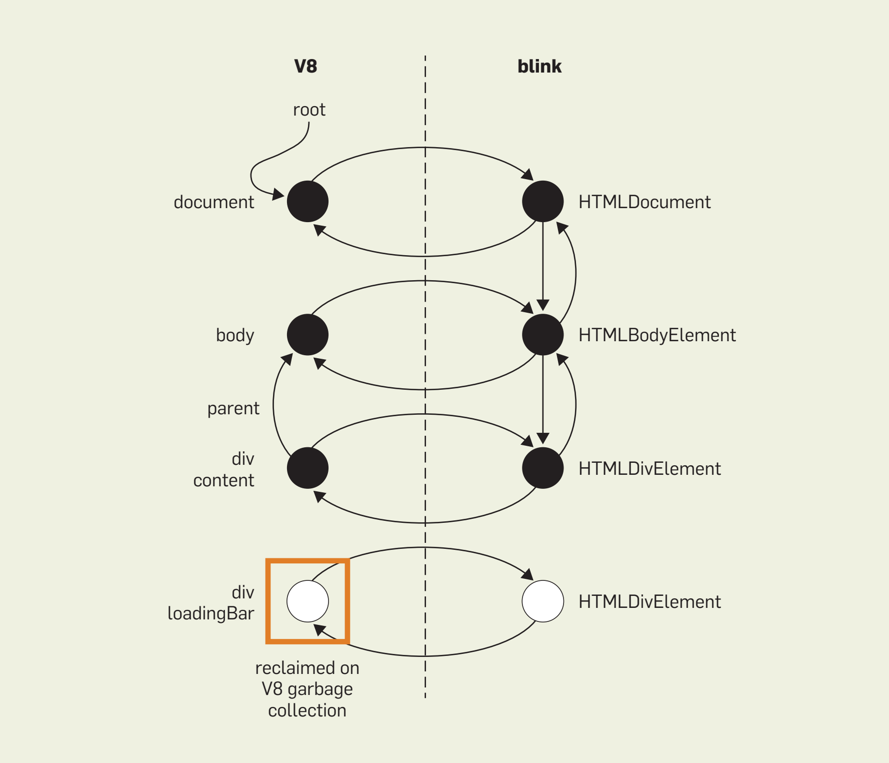
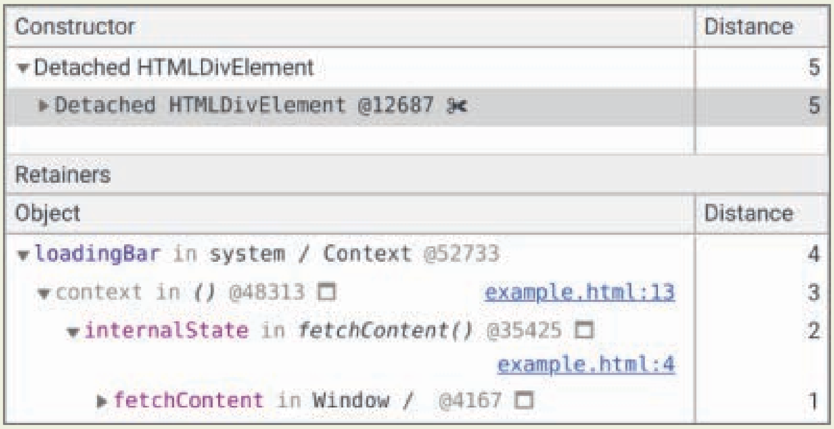


Figure 4. Leaking the callback.

```
function fetchContent(callback) {
        // Emulate network request and content creation.
        setTimeout(callback, 1000);
        fetchContent.internalState = callback;
}
```

Figure 5. Retaining path of the leaking DIV element.



Constructor	Distance
▾Detached HTMLDivElement	5
▸Detached HTMLDivElement @12687	5

Retainers

Object	Distance
▾loadingBar in system / Context @52733	4
▾context in () @48313 □ example.html:13	3
▾internalState in fetchContent() @35425 □ example.html:4	2
▸fetchContent in Window / @4167 □	1

In Chrome, CCT replaced its predecessor, called *object grouping*, in version 57. Object grouping was based on over-approximating liveness across component boundaries by keeping all wrappers and wrappables alive in a given DOM tree as long as a single wrapper was held alive through JavaScript. This assumption was reasonable at the time it was implemented, when modification of the DOM from wrappers occurred infrequently. However, the over-approximation had two major shortcomings: It kept more memory alive than needed, which in times of ever-growing Web applications increased already strong memory pressure in the browser; and, the original algorithm was not designed for incremental processing, which, compared with CCT, resulted in longer garbage-collection pause times.

Incremental CCT as implemented today in Chrome eliminates those problems by providing a much better approximation by computing liveness of objects through reachability and by enabling incremental processing. The detailed performance analysis can be found in the main research paper.[3] We are currently working on concurrent marking of the Blink C++ heap and on integrating CCT into such a scheme.

Debugging

Memory-leak bugs are a widespread problem haunting Web applications today.[7] Powerful language constructs such as closures make it easy for a Web developer to accidentally extend the lifetimes of JavaScript and DOM objects, resulting in higher memory usage than necessary. As a concrete example, let's assume that the fetchContent function from Figure 1 keeps, perhaps because of a bug, an internal reference to the provided callback, as shown in Figure 4.

Without knowing the implementation of the fetchContent function, a Web developer observes that the loadingBar element from the previous example is not reclaimed by the garbage collector. Can debugging tools help track down why the element is leaking?

The tracing infrastructure needed for cross-component garbage collection can be applied to improve memory debugging. Chrome DevTools uses the infrastructure to capture and visualize the object graph spanning JavaScript and DOM objects. The tool allows Web developers to query why a particular object is not reclaimed by the garbage collector. It presents the answer in the form of a *retaining path*, which runs from the object to the garbage-collection root. Figure 5 shows the retaining path for the leaking loadingBar element. The path shows that the leaking DOM element is captured by the loadingBar variable in the environment (called *context* in V8) of an anonymous closure, which is retained by the internalState field of the fetchContent function. By inspecting each node of the path, the Web developer can pinpoint the source of the leak. Thanks to the cross-component tracing, the path seamlessly crosses the DOM and JavaScript boundary.[4]

Reclaiming Memory in Other Heterogeneous Systems

Web browsers are particularly interesting systems, as all major browser engines separate DOM and JavaScript objects in a similar way (that is, by providing different heaps for those objects). Similar to Blink and V8, all those browsers encode their DOM in C++ and must rely on a custom object model for JavaScript. All Blink-derived systems (for example, Chrome, Opera, and Electron) rely on CCT to handle cross-component references. The Gecko rendering engine that powers Firefox uses reference counting to manage DOM objects. An additional incremental cycle collector[1] that wakes up periodically ensures that such cycles are eventually collected. WebKit, the engine running inside Safari, uses reference counting for the C++ DOM with an additional system that computes liveness across the wrapper/wrappable boundary in the final pause of a garbage-collection cycle. Unsurprisingly, all major browsers have mechanisms to deal with these kinds of cycles, as memory leaks in longer-running websites would otherwise be inevitable and would observably impact browser performance.

More interestingly, though, we are not aware of other sophisticated systems integrating VMs that provide cross-component memory management. While VMs often provide bridges for integration in other systems, such as Java Native Interface (JNI) and NativeScript, cross-component references require manual management in all of them. Developers using those systems must manually create and destroy links that can form cycles. This is error prone and can lead to the aforementioned problems.

Conclusion

Cross-component tracing is a way to solve the problem of reference cycles across component boundaries. This problem appears as soon as components can form arbitrary object graphs with nontrivial ownership across API boundaries. An incremental version of CCT is implemented in V8 and Blink, enabling effective and efficient reclamation of memory in a safe manner—without introducing dangling pointers that could lead to program crashes or security vulnerabilities in Chrome or Chromium-derived browsers. The same tracing system is reused by Chrome DevTools to visualize retaining paths of objects independent of whether they are managed in C++ or JavaScript.

Note, however, that CCT comes with significant implementation overhead, as it requires implementations of tracers in each component. Ultimately, implementers need to weigh the effort of either avoiding cycles by enforcing restrictions on their systems or implementing a mechanism to reclaim cycles, such as CCT. Chrome was already equipped with garbage collectors in V8 and Blink, and thus we chose to implement a generic solution such as CCT that allows the systems on top to stay as flexible as needed.

CCT is implemented not only in Chrome, but also in other software systems that use V8 and Chrome, such as the popular Opera Web browser and Electron. Cobalt, a high-performance, small-footprint platform providing a subset of HTML5, CSS, and JavaScript used for embedded devices such as TVs, implemented cross-component tracing inspired by our system to manage its memory.

Related articles on queue.acm.org

Idle-Time Garbage-Collection Scheduling
Ulan Degenbaev et al.
https://queue.acm.org/detail.cfm?id=2977741

Real-time Garbage Collection
David F. Bacon
https://queue.acm.org/detail.cfm?id=1217268

Leaking Space
Neil Mitchell
https://queue.acm.org/detail.cfm?id=2538488

References

1. Bacon, D.F. and Rajan, V.T. Concurrent cycle collection in reference counted systems. In *Proceedings of the 15th European Conf. Object-Oriented Programming*. Springer-Verlag, London, U.K., 2001, 207–235; https://doi.org/10.1007/3-540-45337-7_12.
2. Chambers, C., Ungar, D. and Lee, E. An efficient implementation of SELF, a dynamically-typed object-oriented language based on prototypes. In Proceedings of the Conf. Object-Oriented Programming Systems, Languages and Applications. ACM SIGPLAN, 1989, 49–70; https://dl.acm.org/citation.cfm?doid=74877.74884.
3. Degenbaev, U. et al. Cross-component garbage collection. In Proceedings of the ACM on Programming Languages 2, OOPSLA Article 151, 2018; https://dl.acm.org/citation.cfm?doid=3288538.3276521.
4. Degenbaev, U., Filippov, A., Lippautz, M. and Payer, H. Tracing from JS to the DOM and back again. V8, 2018; https://v8.dev/blog/tracing-js-dom.
5. Degenbaev, U., Lippautz, M. and Payer, H. Concurrent marking in V8. V8, 2018; https://v8.dev/blog/concurrent-marking.
6. Dijkstra, E.W., Lamport, L., Martin, A.J., Scholten, C.S. and Steffens, E.F.M. On-the-fly garbage collection: An exercise in cooperation. *Commun. ACM 21*, 11 (Nov. 1978), 966–975; https://dl.acm.org/citation.cfm?doid=359642.359655.
7. Hablich, M. and Payer, H. Lessons learned from the memory roadshow; https://bit.ly/2018-memory-roadshow.
8. Jones, R., Hosking, A. and Moss, E. *The Garbage Collection Handbook: The Art of Automatic Memory Management*. Chapman & Hall, 2012.

Ulan Degenbaev is a software engineer at Google, working on the garbage collector of the V8 JavaScript engine.

Michael Lippautz is a software engineer at Google, where he works on garbage collection for the V8 JavaScript virtual machine and the Blink rendering engine. Previously, he worked on Google's Dart virtual machine.

Hannes Payer is a software engineer at Google, where he works on the V8 JavaScript virtual machine. Previously, he worked on Google's Dart virtual machine and various Java virtual machines.

DOI:10.1145/3316778

Article development led by acmqueue
queue.acm.org

Build safety, share vulnerability, and establish purpose.

BY KATE MATSUDAIRA

How to Create a Great Team Culture (and Why It Matters)

IN MY CAREER leading teams, I have worked with large organizations (more than 1,000 people) and super-small teams (a startup with just two people). I have seen that the best teams have one thing in common: a strong team culture.

We all know what it is like to be a part of a great team—when you enjoy coming together and the energy is electric. There is something special that happens when the team becomes greater than the sum of the individuals.

I was really inspired by this topic recently when I read Daniel Coyle's book *The Culture Code*.[1] The author shares a lot of research (and I do love data) about what makes a great team. He boils it down to a few key elements:

- *Build safety.* Create an environment where people feel safe and secure.
- *Share vulnerability.* When people are willing to take risks, it can drive cooperation and build trust.
- *Establish purpose.* The team should align around common goals and values, with a clear path forward.

The book is filled with many examples and ideas, but in my experience, I have seen that what works for one team will not work for another. That is one of the reasons leadership is complex and difficult.

You are always working with different variables—different teams, different companies, different goals. And yet team culture is one part of the job that great leaders never ignore. So, how do the best leaders create team culture wherever they go?

See the Role You Play in Team Culture

As a leader, it is your responsibility to set the culture for the team. I am sure you have heard the phrase "lead by example," and that is because when people aren't sure what is acceptable, they look to their leaders for guidance.

You have surely been in the situation where you have seen your manager staying late at the office, and as a result, you might have stayed just a little longer. On the other hand, if you frequently saw your boss taking two-hour lunches, you might not be in such a hurry to get back to the office when your friend stops by to go to lunch.

Every day, people are looking for signals in their environment about what is the norm. As a leader, it is part of your job to set the example for those around you.

You want to create a culture where people are engaged, cooperative, and excited. To do this, you need to be deliberate in your actions. For example, if you want to create a culture of psychological safety, where people can speak up and take risks, it is important that you do not accept or participate in negativity. Research has shown that one bad apple or toxic employee can bring

IMAGE BY ANTON JANKOVOY

down the whole team.[2] As a result, if you see or hear someone acting in opposition to the attitude and environment you are trying to create, you should do your best to diffuse the situation and address the individual quickly.

If you ignore your team culture or think it's not an important part of your job, some type of culture will still develop. That is just what happens when humans work together and share a space. The leader's job is to cultivate the type of culture that will lead to success.

Creating Connections

I was once on a leadership team that had a very negative dynamic—the manager had a tendency to play favorites, so many of my peers were always trying to win favor by badmouthing or undermining the others.

I remember being very frustrated that this was allowed to happen and decided that whenever someone came to me to complain about another team member, instead of sharing that feedback, I would coach the complainer to tell the other person directly. If the complaint was about how a peer was managing a project, I would talk about how that person could improve and come up with a plan for the complainer to help that other person succeed.

At that time, I was reacting to the problems I saw, but in retrospect, I realized my actions ended up creating a cohesive team—one where people were encouraged to help one another, and if they disagreed, would always try to sort it out themselves.

A leader should help create mutual vulnerability among team members. This can be done in a few ways:

- *Force collaboration.* Have team members work together to solve a problem or complete a project.
- *Encourage people to talk to one another directly.* Foster an environment of peer coaching and resist being a proxy for critical feedback.
- *Reward and recognize collaboration.* Drive for shared outcomes that celebrate team successes.
- *Build trust.* Create opportunities (for example, summits or meetings) for people to build trust with one another and get to know each another as people—not just as coworkers.

There are many more strategies, but the key is to create opportunities where people can connect. Help build trust among team members by allowing them to resolve their own conflicts without you being the mediator. Over time, this will create a group of people who can trust one another, which, in turn, will create a cohesive team culture.

Define Your Culture by Knowing Its Value

Defining a team culture is not an easy job. Once a culture has been established, it is easy to see its signposts. But when you are new to a team or working to develop a positive team culture, it can be challenging to know exactly what it takes to build a culture.

Before you start picking rituals and values you think "sound" good, start by going back to the very beginning. Ask yourself: *What is the value of this team?* Really think about it. *Why does this team exist?* Think beyond the function that the team serves for the organization as a whole (for example, coding or design). *Why are we a team?* The answer to this question is not always clear.

Other questions to consider: What are the advantages of being a team? What can we do because we are working together? Maybe it is so team members can learn from each other and do better work; there is more learning when there is collaboration. Maybe it is decision making; we make smarter decisions when we have multiple viewpoints. Maybe it is about resources; we can do more with combined skills and time.

How to Create Cultural Touchpoints Around Your Values

Once you know the value of your team, then you can start building in elements that support its values.

Let's say you decide that one of the values of your team is peer mentoring; in other words, one of the reasons your team exists is so that its members can do better work by learning from one another. Now, how do you make that happen on your team?

You cannot just say "We learn from each other" during a meeting and make it happen. You have to institute processes that make this a simple part of daily life on your team. Think about which kinds of forums you can set up to help people learn from one another. Do you want to encourage questions in team chat? Set up a code-review process? Establish a cadence of brown bags to share lessons learned? Read whitepapers and discuss them as a group? What makes sense for your team?

Now let's say that another value of your team is decision-making; your team exists because everyone's input helps to make smarter decisions about what to work on and how to work on it. We all know that the more people are involved in a decision, the more complicated it becomes to get a clear answer. So, how do you benefit from getting everyone's input without becoming a team that can get nothing done because no one can agree?

The solution might be to have all decision-making done in the same way. For example, you might institute a format for all reports. That way, the input from various individuals or departments will all come to you in the exact same way, so you can quickly parse the information and make a decision.

Team Structure Becomes Team Culture

Team culture is not just wearing the same t-shirts at the company picnic. The way you do things every day is what builds your culture.

So, while streamlining the way your team formats their reports might not feel like it has much to do with team culture, it does. It is a way of steering your team to work together, by prioritizing the values that your team supports.

Culture is in the everyday. It is the small actions that you and everyone on your team takes on a daily basis—the way they speak to each other, the way decisions get made, the way they run meetings—that make up your team culture.

I have seen many amazing examples of culture-building throughout my career. Here are just a few more ideas that might inspire you as you build your team culture:

- *Weekly demo meetings.* Have someone from the team share a recent accomplishment. This could be a big thing, or even something as small as changing a button color on the website. This creates a culture of sharing work so that people feel more collaborative even outside the meeting setting, since they know what other people are working on.
- *Teaching slots for every team member.* At every team meeting, have people sign up to share something they've learned recently or teach something to the team. This is a great way for people who have been to conferences or read interesting books to share that knowledge, and it helps give everyone on the team a voice (even those who don't normally speak up during meetings).
- *Cupcakes for launches.* Mark every team win by bringing people together—literally together, around a plate of cupcakes, instead of just via email. This creates a culture of celebration, where people's successes are noticed and rewarded, and where the whole team celebrates together.

Some of the decisions you make will feel big and some will feel small. But whether it's as huge as developing an online help tool for your team or as small as sharing cupcakes after a big win, the effect is the same. Culture comes from shared experiences. The *what* doesn't matter nearly as much as the *why*.

How can you make people feel like they are valued and important parts of the team?

Let the Culture Expand From the Top Down

As leader of the team, you have significant influence over your team's culture. You can institute policies and procedures that help make your team happy and productive, monitor team successes, and continually improve the team.

Another important part of team culture, however, is helping people feel they are a part of creating it. How can you expand the job of creating a culture to other team members?

Look for opportunities to delegate whenever you can. If a holiday is coming up, maybe you could ask a team member to help organize a team dinner. Look for people with unique perspectives (who maybe aren't heard from as often as others) and give them a platform to share.

This is where truly great leadership comes from. You establish a culture that enables your team to be the best it can be, and then you allow the team to take that culture and run with it.

How amazing could your team be with just a few adjustments? C

Related articles on queue.acm.org

High-Performance Team
Philip Beevers
https://queue.acm.org/detail.cfm?id=1117402

Culture Surprises in Remote Software Development Teams
Judith S. Olson and Gary M. Olson
https://queue.acm.org/detail.cfm?id=966804

Stand and Deliver: Why I Hate Stand-Up Meetings
Phillip A. Laplante
https://queue.acm.org/detail.cfm?id=957730

References

1. Coyle, D. *The Culture Code.* Bantam, 2018; http://danielcoyle.com/the-culture-code/.
2. Felps, W., Mitchell, T., and Byington, E. How, when, and why bad apples spoil the barrel: Negative group members and dysfunctional groups. *Research in Organizational Behavior 27* (2006), 175–222.

Kate Matsudaira (katemats.com) is an experienced technology leader. She has worked at Microsoft and Amazon and successful startups before starting her own company, Popforms, which was acquired by Safari Books.

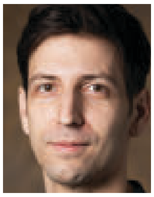
DOI:10.1145/3316774

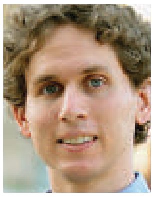
Article development led by acmqueue
queue.acm.org

Some ML papers suffer from flaws that could mislead the public and stymie future research.

BY ZACHARY C. LIPTON AND JACOB STEINHARDT

Research for Practice: Troubling Trends in Machine-Learning Scholarship





COLLECTIVELY, MACHINE LEARNING (ML) researchers are engaged in the creation and dissemination of knowledge about data-driven algorithms. In a given paper, researchers might aspire to any subset of the following goals, among others: to theoretically characterize what is learnable; to obtain understanding through empirically rigorous experiments; or to build a working system that has high predictive accuracy. While determining which knowledge warrants inquiry may be subjective, once the topic is fixed, papers are most valuable to the community when they act in service of the reader, creating foundational knowledge and communicating as clearly as possible.

What sorts of papers best serve their readers? Ideally, papers should accomplish the following: provide intuition to aid the reader's understanding but clearly distinguish it from stronger conclusions supported by evidence; describe empirical investigations that consider and rule out alternative hypotheses; make clear the relationship between theoretical analysis and intuitive or empirical claims; and use language to empower the reader, choosing terminology to avoid misleading or unproven connotations, collisions with other definitions, or conflation with other related but distinct concepts.

Recent progress in machine learning comes despite frequent departures from these ideals. This installment of Research for Practice focuses on the following four patterns that appear to be trending in ML scholarship:

- Failure to distinguish between explanation and speculation.
- Failure to identify the sources of empirical gains (for example, emphasizing unnecessary modifications to neural architectures when gains actually stem from hyperparameter tuning).
- "Mathiness"—the use of mathematics that obfuscates or impresses rather than clarifies (for example, by confusing technical and nontechnical concepts).
- Misuse of language (for example, by choosing terms of art with colloquial connotations or by overloading established technical terms).

While the causes of these patterns are uncertain, possibilities include the rapid expansion of the community, the consequent thinness of the reviewer pool, and the often-misaligned incentives between scholarship and short-term measures of success (for example, bibliometrics, attention, and entrepreneurial opportunity). While each pattern offers a corresponding remedy (don't do it), this article also makes suggestions on how the community might combat these troubling trends.

As the impact of machine learning widens, and the audience for research papers increasingly includes students, journalists, and policy-makers, these considerations apply to this wider audience as well. By communicating more precise information with greater clarity, better ML scholarship could accelerate the pace of research, reduce the on-boarding time for new researchers, and play a more constructive role in public discourse.

Flawed scholarship threatens to mislead the public and stymie future research by compromising ML's intellectual foundations. Indeed, many of these problems have recurred cyclically throughout the history of AI (artificial intelligence) and, more broadly, in scientific research. I n 1976, Drew McDermott[26] chastised the AI community for abandoning self-discipline, warning prophetically "if we can't criticize ourselves, someone else will save us the trouble." Similar discussions recurred throughout the 1980s, 1990s, and 2000s. In other fields, such as psychology, poor experimental standards have eroded trust in the discipline's authority.[33] The current strength of machine learning owes to a large body of rigorous research to date, both theoretical and empirical. By promoting clear scientific thinking and communication, our community can sustain the trust and investment it currently enjoys.

Disclaimers. This article aims to instigate discussion, answering a call for papers from the International Conference on Machine Learning (ICML) Machine Learning Debates workshop. While we stand by the points represented here, we do not purport to offer a full or balanced viewpoint or to discuss the overall quality of science in ML. In many aspects, such as reproducibility, the community has advanced standards far beyond what sufficed a decade ago.

Note that these arguments are made by *us*, against *us*—insiders offering a critical introspective look—not as sniping outsiders. The ills identified here are not specific to any individual or institution. We have fallen into these patterns ourselves, and likely will again in the future. Exhibiting one of these patterns doesn't make a paper *bad*, nor does it indict the paper's authors; however, all papers could be made stronger by avoiding these patterns.

While we provide concrete examples, our guiding principles are to implicate ourselves; and to select preferentially from the work of better-established researchers and institutions that we admire, to avoid singling out junior students for whom inclusion in this discussion might have consequences and who lack the opportunity to reply symmetrically. We are grateful to belong to a community that provides sufficient intellectual freedom to allow the expression of critical perspectives.

Troubling Trends

Each subsection that follows describes a trend; provides several examples (as well as positive examples that resist the trend); and explains the consequences. Pointing to weaknesses in individual papers can be a sensitive topic. To minimize this, the examples are short and specific.

Explanation vs. speculation. Research into new areas often involves exploration predicated on intuitions that have yet to coalesce into crisp formal representations. Speculation is a way for authors to impart intuitions that may not yet withstand the full weight of scientific scrutiny. Papers often offer speculation in the guise of *explanations*, however, which are then interpreted as authoritative because of the trappings of a scientific paper and the presumed expertise of the authors.

For instance, in a 2015 paper, Ioffe and Szegedy[18] form an intuitive theory around a concept called *internal covariate shift*. The exposition on internal covariate shift, starting from the abstract, appears to state technical facts. Key terms are not made crisp enough, however, to assume a truth value conclusively. For example, the paper states that batch normalization offers improvements by reducing changes in the distribution of hidden activations over the course of training. By which divergence measure is this change quantified? The paper never clarifies, and some work suggests that this explanation of batch normalization may be off the mark.[37] Nevertheless, the speculative explanation given by Ioffe and Szegedy has been repeated as fact—for example, in a 2015 paper by Noh, Hong, and Han,[31] which states, "It is well known that a deep neural network is very hard to optimize due to the internal-covariate-shift problem."

We have been equally guilty of speculation disguised as explanation. In a 2017 paper with Koh and Liang,[42] I (Jacob Steinhardt) wrote that "the high dimensionality and abundance of irrelevant features ... give the attacker more room to construct attacks," without conducting any experiments to measure the effect of dimensionality on attackability. In another paper with Liang from 2015,[41] I (Steinhardt) introduced the intuitive notion of *coverage* without defining it, and used it as a form of explanation (for example, "Recall that one symptom of a lack of coverage is poor estimates of uncertainty and the inability to generate high-precision predictions." Looking back, we desired to communicate insufficiently fleshed-out intuitions that were material to the work described in the paper and were reticent to label a core part of the argument as speculative.

In contrast to these examples, Srivastava et al.[39] separate speculation from fact. While this 2014 paper, which introduced dropout regularization, speculates at length on connections between dropout and sexual reproduction, a designated "Motivation" section clearly quarantines this discussion. This practice avoids confusing readers while allowing authors to express informal ideas.

In another positive example, Yoshua Bengio[2] presents practical guidelines for training neural networks. Here, the author carefully conveys uncertainty. Instead of presenting the guidelines as authoritative, the paper states: "Although such recommendations come … from years of experimentation and to some extent mathematical justification, they should be challenged. They constitute a good starting point … but very often have not been formally validated, leaving open many questions that can be answered either by theoretical analysis or by solid comparative experimental work."

Failure to identify the sources of empirical gains. The ML peer-review process places a premium on technical novelty. Perhaps to satisfy reviewers, many papers emphasize both complex models (addressed here) and fancy mathematics (to be discussed in "Mathiness" section). While complex models are sometimes justified, empirical advances often come about in other ways: through clever problem formulations, scientific experiments, optimization heuristics, data-preprocessing techniques, extensive hyperparameter tuning, or applying existing methods to interesting new tasks. Sometimes a number of proposed techniques together achieve a significant empirical result. In these cases, it serves the reader to elucidate which techniques are necessary to realize the reported gains.

Too frequently, authors propose many tweaks absent proper ablation studies, obscuring the source of empirical gains. Sometimes, just one of the changes is actually responsible for the improved results. This can give the false impression that the authors did more work (by proposing several improvements), when in fact they did not do enough (by not performing proper ablations). Moreover, this practice misleads readers to believe that all of the proposed changes are necessary.

Empirical study aimed at understanding can be illuminating even absent a new algorithm.

In 2018, Melis, Dyer, and Blunsom[27] demonstrated that a series of published improvements in language modeling, originally attributed to complex innovations in network architectures, were actually the result of better hyperparameter tuning. On equal footing, vanilla long short-term memory (LSTM) networks, hardly modified since 1997, topped the leaderboard. The community might have benefited more by learning the details of the hyperparameter tuning without the distractions. Similar evaluation issues have been observed for deep reinforcement learning[17] and generative adversarial networks.[24] See Sculley et al.[38] for more discussion of lapses in empirical rigor and resulting consequences.

In contrast, many papers perform good ablation analyses, and even retrospective attempts to isolate the source of gains can lead to new discoveries. Furthermore, ablation is neither necessary nor sufficient for understanding a method, and can even be impractical given computational constraints. Understanding can also come from robustness checks (as in Cotterell et al.,[9] which discovers that existing language models handle inflectional morphology poorly), as well as qualitative error analysis.

Empirical study aimed at understanding can be illuminating even absent a new algorithm. For example, probing the behavior of neural networks led to identifying their susceptibility to adversarial perturbations.[44] Careful study also often reveals limitations of challenge datasets while yielding stronger baselines. A 2016 paper by Chen, Bolton, and Manning[6] studied a task designed for reading comprehension of news passages and found that 73% of the questions can be answered by looking at a single sentence, while only 2% required looking at multiple sentences (the remaining 25% of examples were either ambiguous or contained coreference errors). In addition, simpler neural networks and linear classifiers outperformed complicated neural architectures that had previously been evaluated on this task. In the same spirit, Zellers et al.[45] analyzed and constructed a strong baseline for the Visual Genome Scene Graphs dataset in their 2018 paper.

Mathiness. When writing a paper early in my Ph.D. program, I (Zachary Lipton) received feedback from an experienced post-doc that the paper needed more equations. The post-doc wasn't endorsing the system but rather communicating a sober view of how reviewing works. More equations, even when difficult to decipher, tend to convince reviewers of a paper's technical depth.

Mathematics is an essential tool for scientific communication, imparting precision and clarity when used correctly. Not all ideas and claims are amenable to precise mathematical description, however, and natural language is an equally indispensable tool for communicating, especially about intuitive or empirical claims.

When mathematical and natural-language statements are mixed without a clear accounting of their relationship, both the prose and the theory can suffer: problems in the theory can be concealed by vague definitions, while weak arguments in the prose can be bolstered by the appearance of technical depth. We refer to this tangling of formal and informal claims as mathiness, following economist Paul Romer, who described the pattern like this: "Like mathematical theory, *mathiness* uses a mixture of words and symbols, but instead of making tight links, it leaves ample room for slippage between statements in natural language versus formal language."[36]

When mathematical and natural-language statements are mixed without a clear accounting of their relationship, both the prose and the theory can suffer.

Mathiness manifests in several ways. First, some papers abuse mathematics to convey technical depth—to bulldoze rather than to clarify. Spurious theorems are common culprits, inserted into papers to lend authoritativeness to empirical results, even when the theorem's conclusions do not actually support the main claims of the paper. I (Steinhardt) was guilty of this in a 2015 paper with Percy Liang,[40] where a discussion of "staged strong Doeblin chains" had limited relevance to the proposed learning algorithm but might confer a sense of theoretical depth to readers.

The ubiquity of this issue is evidenced by the paper introducing the Adam optimizer.[19] In the course of introducing an optimizer with strong empirical performance, it also offers a theorem regarding convergence in the convex case, which is perhaps unnecessary in an applied paper focusing on non-convex optimization. The proof was later shown to be incorrect.[35]

A second mathiness issue is putting forth claims that are neither clearly formal nor clearly informal. For example, Dauphin et al.[11] argued that the difficulty in optimizing neural networks stems not from local minima but from saddle points. As one piece of evidence, the work cites a statistical physics paper by Bray and Dean[5] on Gaussian random fields and states that in high dimensions "all local minima [of Gaussian random fields] are likely to have an error very close to that of the global minimum." (A similar statement appears in the related work of Choromanska et al.[7]) This appears to be a formal claim, but absent a specific theorem it is difficult to verify the claimed result or to determine its precise content. Our understanding is that it is partially a numerical claim that the gap is small for typical settings of the problem parameters, as opposed to a claim that the gap vanishes in high dimensions. A formal statement would help clarify this. Note that the broader interesting point in Dauphin et al. that minima tend to have lower loss than saddle points is more clearly stated and empirically tested.

Finally, some papers invoke theory in overly broad ways or make passing references to theorems with dubious pertinence. For example, the no-free-lunch theorem is commonly invoked as a justification for using heuristic methods without guarantees, even though the theorem does not formally preclude guaranteed learning procedures.

While the best remedy for mathiness is to avoid it, some papers go further with exemplary exposition. A 2013 paper by Bottou et al.[4] on counterfactual reasoning covered a large amount of mathematical ground in a down-to-earth manner, with numerous clear connections to applied empirical problems. This tutorial, written in clear service to the reader, has helped to spur work in the burgeoning community studying counterfactual reasoning for ML.

Misuse of language. There are three common avenues of language misuse in machine learning: suggestive definitions, overloaded terminology, and suitcase words.

Suggestive definitions. In the first avenue, a new technical term is coined that has a suggestive colloquial meaning, thus sneaking in connotations without the need to argue for them. This often manifests in anthropomorphic characterizations of tasks (*reading comprehension* and *music composition*) and techniques (*curiosity* and *fear*—I (Zachary) am responsible for the latter). A number of papers name components of proposed models in a manner suggestive of human cognition (for example, *thought vectors* and the *consciousness prior*). Our goal is not to rid the academic literature of all such language; when properly qualified, these connections might communicate a fruitful source of inspiration. When a suggestive term is assigned technical meaning, however, each subsequent paper has no choice but to confuse its readers, either by embracing the term or by replacing it.

Describing empirical results with loose claims of "human-level" performance can also portray a false sense of current capabilities. Take, for example, the "dermatologist-level classification of skin cancer" reported in a 2017 paper by Esteva et al.[12] The comparison with dermatologists concealed the fact that classifiers and dermatologists perform fundamentally different tasks. Real dermatologists encounter a wide variety of circumstances and must perform their jobs despite unpredictable changes. The machine classifier, however, achieveed low error only on independent, identically distributed (IID) test data.

In contrast, claims of human-level performance in work by He et al.[16] are better qualified to refer to the ImageNet classification task (rather than object recognition more broadly). Even in this case, one careful paper (among many less careful) was insufficient to put the public discourse back on track. Popular articles continue to characterize modern image classifiers as "surpassing human abilities and effectively proving that bigger data leads to better decisions," as explained by Dave Gershgorn,[13] despite demonstrations that these networks rely on spurious correlations, (for example, misclassifying "Asians dressed in red" as ping-pong balls, reported by Stock and Cisse[43]).

Deep-learning papers are not the sole offenders; misuse of language plagues many subfields of ML. Lipton, Chouldechova, and McAuley[23] discuss how the recent literature on fairness in ML often overloads terminology borrowed from complex legal doctrine, such as *disparate impact*, to name simple equations expressing particular notions of statistical parity. This has resulted in a literature where "fairness," "opportunity," and "discrimination" denote simple statistics of predictive models, confusing researchers who become oblivious to the difference and policymakers who become misinformed about the ease of incorporating ethical desiderata into ML.

Overloading technical terminology. A second avenue of language misuse consists of taking a term that holds precise technical meaning and using it in an imprecise or contradictory way. Consider the case of *deconvolution*, which formally describes the process of reversing a convolution, but is now used in the deep-learning literature to refer to *transpose convolutions* (also called *upconvolutions*) as commonly found in auto-encoders and generative adversarial networks. This term first took root in deep learning in a paper that does address deconvolution but was later overgeneralized to refer to any neural architecture using upconvolutions. Such overloading of terminology can create lasting confusion. New ML papers referring to deconvolution might be invoking its original meaning, describing upconvolution, or attempting to resolve the confusion, as in a paper by Hazirbas, Leal-Taixé, and Cremers,[15] which awkwardly refers to "upconvolution (deconvolution)."

As another example, *generative models* are traditionally models of either the input distribution $p(x)$ or the joint distribution $p(x,y)$. In contrast, discriminative models address the conditional distribution $p(y|x)$ of the label given the inputs. In recent works, however, *generative model* imprecisely refers to any model that produces realistic-looking structured data. On the surface, this may seem consistent with the $p(x)$ definition, but it obscures several shortcomings—for example, the inability of GANs (generative adversarial networks) or VAEs (variational autoencoders) to perform conditional inference (for example, sampling from $p(x_2|x_1)$ where x_1 and x_2 are two distinct input features). Bending the term further, some discriminative models are now referred to as generative models on account of producing structured outputs, a mistake that I (Lipton), too, have made. Seeking to resolve the confusion and provide historical context, Mohamed and Lakshminarayanan[30] distinguish between *prescribed* and *implicit* generative models.

Revisiting batch normalization, Ioffe and Szegedy[18] described *covariate shift* as a change in the distribution of model inputs. In fact, *covariate shift* refers to a specific type of shift, where although the input distribution $p(x)$ might change, the labeling function $p(y|x)$ does not. Moreover, as a result of the influence of Ioffe and Szegedy, Google Scholar lists batch normalization as the first reference on searches for "covariate shift."

Among the consequences of misusing language is the possibility (as with generative models) of concealing lack of progress by redefining an unsolved task to refer to something easier. This often combines with suggestive definitions via anthropomorphic naming. *Language understanding* and *reading comprehension*, once grand challenges of AI, now refer to making accurate predictions on specific datasets.

Suitcase words. Finally, ML papers tend to overuse suitcase words. Coined by Marvin Minsky in the 2007 book *The Emotion Machine*,[29] suitcase words pack together a variety of meanings. Minsky described mental processes such as consciousness, thinking, attention, emotion, and feeling that may not share "a single cause or origin." Many terms in ML fall into this category. For example, I (Lipton) noted in a 2016 paper that *interpretability* holds no universally agreed-upon meaning and often references disjoint methods and desiderata.[22] As a consequence, even papers that appear to be in dialogue with each other may have different concepts in mind.

As another example, *generalization* has both a specific technical meaning (generalizing from training to testing) and a more colloquial meaning that is closer to the notion of transfer (generalizing from one population to another) or of external validity (generalizing from an experimental setting to the real world). Conflating these notions leads to overestimating the capabilities of current systems.

Suggestive definitions and overloaded terminology can contribute to the creation of new suitcase words. In the fairness literature, where legal, philosophical, and statistical language are often overloaded, terms such as *bias* become suitcase words that must be subsequently unpacked.

In common speech and as aspirational terms, suitcase words can serve a useful purpose. Sometimes a suitcase word might reflect an overarching aspiration that unites the various meanings. For example, *artificial intelligence* might be well suited as an aspirational name to organize an academic department. On the other hand, using suitcase words in technical arguments can lead to confusion. For example, in his 2017 book, *Superintelligence*,[3] Nick Bostrom wrote an equation (Box 4) involving the terms *intelligence* and *optimization power*, implicitly assuming these suitcase words can be quantified with a one-dimensional scalar.

Speculation on Causes Behind the Trends

Do the patterns mentioned here represent a trend, and if so, what are the underlying causes? We speculate that these patterns are on the rise and suspect several possible causal factors: complacency in the face of progress, the rapid expansion of the community, the consequent thinness of the reviewer pool, and misaligned incentives of scholarship vs. short-term measures of success.

Complacency in the face of progress. The apparent rapid progress in ML has at times engendered an attitude that *strong results excuse weak arguments*. Authors with strong results may feel licensed to insert arbitrary unsupported stories (see "Explanation vs. Speculation") regarding the factors driving the results; to omit experiments aimed at disentangling those factors (see "Failure to Identify the Sources of Empirical Gains"); to adopt exaggerated terminology (see "Misuse of Language"); or to take less care to avoid mathiness (see "Mathiness").

At the same time, the single-round nature of the reviewing process may cause reviewers to feel they have no choice but to accept papers with strong quantitative findings. Indeed, even if the paper is rejected, there is no guarantee the flaws will be fixed or even noticed in the next cycle, so reviewers may conclude that accepting a flawed paper is the best option.

Growing pains. Since around 2012, the ML community has expanded rapidly because of increased popularity stemming from the success of deep-learning methods. While the rapid expansion of the community can be seen as a positive development, it can also have side effects.

To protect junior authors, we have preferentially referenced our own papers and those of established researchers. And certainly, experienced researchers exhibit these patterns. Newer researchers, however, may be even more susceptible. For example, authors unaware of previous terminology are more likely to misuse or redefine language (as discussed earlier).

Rapid growth can also thin the reviewer pool in two ways: by increasing the ratio of submitted papers to reviewers and by decreasing the fraction of experienced reviewers. Less-experienced reviewers may be more likely to demand architectural novelty, be fooled by spurious theorems, and let pass serious but subtle issues such as misuse of language, thus either incentivizing or enabling several of the trends described here. At the same time, experienced but overburdened reviewers may revert to a "checklist" mentality, rewarding more formulaic papers at the expense of more creative or intellectually ambitious work that might not fit a preconceived template. Moreover, overworked reviewers may not have enough time to fix—or even to notice—all of the issues in a submitted paper.

Misaligned incentives. Reviewers are not alone in providing poor incentives for authors. As ML research garners increased media attention and ML startups become commonplace, to some degree incentives are provided by the press ("What will they write about?") and by investors ("What will they invest in?"). The media provides incentives for some of these trends.

Anthropomorphic descriptions of ML algorithms provide fodder for popular coverage. Take, for example, a 2014 article by Cade Metz in *Wired*,[28] that characterized an autoencoder as a "simulated brain." Hints of human-level performance tend to be sensationalized in newspaper coverage—for example, an article in the *New York Times* by John Markoff described a deep-learning image-captioning system as "mimicking human levels of understanding."[25]

Investors, too, have shown a strong appetite for AI research, funding startups sometimes on the basis of a single paper. In my (Lipton) experience working with investors, they are sometimes attracted to startups whose research has received media coverage, a dynamic that attaches financial incentives to media attention. Note that recent interest in chatbot startups co-occurred with anthropomorphic descriptions of dialogue systems and reinforcement learners both in papers and in the media, although it may be difficult to determine whether the lapses in scholarship caused the interest of investors or vice versa.

Suggestions. Suppose we are to intervene to counter these trends, then how? Besides merely suggesting that each author abstain from these patterns, what can we do as a community to raise the level of experimental practice, exposition, and theory? And how can we more readily distill the knowledge of the community and disabuse researchers and the wider public of misconceptions? What follows are a number of preliminary suggestions based on personal experiences and impressions.

For Authors, Publishers, and Reviewers

We encourage authors to ask "What worked?" and "Why?" rather than just "How well?" Except in extraordinary cases, raw headline numbers provide limited value for scientific progress absent insight into what drives them. Insight does not necessarily mean theory. Three practices that are common in the strongest empirical papers are error analysis, ablation studies, and robustness checks (for example, choice of hyperparameters, as well as ideally the choice of dataset). Everyone can adopt these practices, and we advocate their widespread use. For some exemplar papers, consider the preceding discussion in "Failure to Identify the Sources of Empirical Gains." Langley and Kibler[21] also provide a more detailed survey of empirical best practices.

Sound empirical inquiry need not be confined to tracing the sources of a particular algorithm's empirical gains; it can yield new insights even when no new algorithm is proposed. Notable examples of this include a demonstration that neural networks trained by stochastic gradient descent can fit randomly assigned labels.[46] This paper questions the ability of learning-theoretic notions of model complexity to explain why neural networks can generalize to unseen data. In another example, Goodfellow, Vinyals, and Saxe[14] explored the loss surfaces of deep networks, revealing that straight-line paths in parameter space between initialized and learned parameters typically have monotonically decreasing loss.

When researchers are writing their papers, we recommend they ask the following question: *Would I rely on this explanation for making predictions or for getting a system to work?* This can be a good test of whether a theorem is being included to please reviewers or to convey actual insight. It also helps check whether concepts and explanations match the researcher's own internal mental model. On mathematical writing, we point the reader to Knuth, Larrabee, and Roberts's excellent guidebook, *Mathematical Writing*.[20]

Finally, being clear about which problems are open and which are solved not only presents a clearer picture to readers, but also encourages follow-up work and guards against researchers neglecting questions presumed (falsely) to be resolved.

Reviewers can set better incentives by asking: "Might I have accepted this paper if the authors had done a worse job?" For example, a paper describing a simple idea that leads to improved performance, together with two negative results, should be judged more favorably than a paper that combines three ideas together (without ablation studies) yielding the same improvement.

Investors have shown a strong appetite for AI research, funding startups sometimes on the basis of a single paper.

Current literature moves fast at the expense of accepting flawed works for conference publication. One remedy could be to emphasize authoritative retrospective surveys that strip out exaggerated claims and extraneous material, change anthropomorphic names to sober alternatives, standardize notation, and so on. While venues such as *Foundations and Trends in Machine Learning*, a journal from Now Publishers in Hanover, MA, already provide a track for such work, there are still not enough strong papers in this genre.

Additionally, we believe (noting our conflict of interest) that critical writing ought to have a voice at ML conferences. Typical ML conference papers choose an established problem (or propose a new one), demonstrate an algorithm and/or analysis, and report experimental results. While many questions can be approached in this way, when addressing the validity of the problems or the methods of inquiry themselves, neither algorithms nor experiments are sufficient (or appropriate). We would not be alone in embracing greater critical discourse: in natural language processing (NLP), this year's Conference on Computational Linguistics (COLING) included a call for position papers "to challenge conventional thinking."

There are many lines of further discussion worth pursuing regarding peer review. Are the problems described here mitigated or exacerbated by open review? How do reviewer point systems align with the values that we advocate? These topics warrant their own papers and have indeed been discussed at length elsewhere.

Discussion. Folk wisdom might suggest not to intervene just as the field is heating up—you can't argue with success! We counter these objections with the following arguments: First, many aspects of the current culture are *consequences* of ML's recent success, not its *causes*. In fact, many of the papers leading to the current success of deep learning were careful empirical investigations characterizing principles for training deep networks. This includes the advantage of random over sequential hyperparameter search, the behavior of different activation functions, and an understanding of unsupervised pretraining.

Second, flawed scholarship already negatively impacts the research community and broader public discourse. The "Troubling Trends" section of this article gives examples of unsupported claims being cited thousands of times, lineages of purported improvements being overturned by simple baselines, datasets that appear to test high-level semantic reasoning but actually test low-level syntactic fluency, and terminology confusion that muddles the academic dialogue. This final issue also affects public discourse. For example, the European Parliament passed a report considering regulations to apply if "robots become or are made self-aware."[10] While ML researchers are not responsible for all misrepresentations of our work, it seems likely that anthropomorphic language in authoritative peer-reviewed papers is at least partly to blame.

Greater rigor in exposition, science, and theory are essential for both scientific progress and fostering productive discourse with the broader public. Moreover, as practitioners apply ML in critical domains such as health, law, and autonomous driving, a calibrated awareness of the abilities and limits of ML systems will help us to deploy ML responsibly.

Greater rigor in exposition, science, and theory are essential for both scientific progress and fostering productive discourse with the broader public.

Countervailing Considerations

There are a number of countervailing considerations to the suggestions set forth in this article. Several readers of earlier drafts of this paper noted that *stochastic gradient descent tends to converge faster than gradient descent*—in other words, perhaps a faster, noisier process that ignores our guidelines for producing "cleaner" papers results in a faster pace of research. For example, the breakthrough paper on ImageNet classification proposes multiple techniques without ablation studies, several of which were subsequently determined to be unnecessary. At the time, however, the results were so significant and the experiments so computationally expensive to run that waiting for ablations to complete might not have been worth the cost to the community.

A related concern is that high standards might impede the publication of original ideas, which are more likely to be unusual and speculative. In other fields, such as economics, high standards result in a publishing process that can take years for a single paper, with lengthy revision cycles consuming resources that could be deployed toward new work.

Finally, perhaps there is value in specialization: The researchers generating new conceptual ideas or building new systems need not be the same ones who carefully collate and distill knowledge.

These are valid considerations, and the standards we are putting forth here are at times exacting. In many cases, however, they are straightforward to implement, requiring only a few extra days of experiments and more careful writing. Moreover, they are being presented as strong heuristics rather than unbreakable rules—if an idea cannot be shared without violating these heuristics, the idea should be shared and the heuristics set aside.

We have almost always found attempts to adhere to these standards to be well worth the effort. In short, the research community has not achieved a Pareto optimal state on the growth-quality frontier.

Historical Antecedents

The issues discussed here are unique neither to machine learning nor to this moment in time; they instead reflect issues that recur cyclically throughout academia. As far back as 1964, the physicist John R. Platt[34] discussed related concerns in his paper on strong inference, where he identified adherence to specific empirical standards as responsible for the rapid progress of molecular biology and high-energy physics relative to other areas of science.

There have been similar discussions in AI. As noted in the introduction to this article, McDermott[26] criticized a (mostly pre-ML) AI community in 1976 on a number of issues, including suggestive definitions and a failure to separate out speculation from technical claims. In 1988, Cohen and

Howe[8] addressed an AI community that at that point "rarely publish[ed] performance evaluations" of their proposed algorithms and instead only described the systems. They suggested establishing sensible metrics for quantifying progress, and analyzing the following: "Why does it work?" "Under what circumstances won't it work?" and "Have the design decisions been justified?"—questions that continue to resonate today.

Finally, in 2009 Armstrong et al.[1] discussed the empirical rigor of information-retrieval research, noting a tendency of papers to compare against the same weak baselines, producing a long series of improvements that did not accumulate to meaningful gains.

In other fields, an unchecked decline in scholarship has led to crisis. A landmark study in 2015 suggested a significant portion of findings in the psychology literature may not be reproducible.[33] In a few historical cases, enthusiasm paired with undisciplined scholarship led entire communities down blind alleys. For example, following the discovery of X-rays, a related discipline on N-rays emerged before it was eventually debunked.[32]

Concluding Remarks

The reader might rightly suggest these problems are self-correcting. We agree. However, the community self-corrects precisely through recurring debate about what constitutes reasonable standards for scholarship. We hope that this paper contributes constructively to the discussion.

Acknowledgments

We thank Asya Bergal, Kyunghyun Cho, Moustapha Cisse, Daniel Dewey, Danny Hernandez, Charles Elkan, Ian Goodfellow, Moritz Hardt, Tatsunori Hashimoto, Sergey Ioffe, Sham Kakade, David Kale, Holden Karnofsky, Pang Wei Koh, Lisha Li, Percy Liang, Julian McAuley, Robert Nishihara, Noah Smith, Balakrishnan "Murali" Narayanaswamy, Ali Rahimi, Christopher Ré, and Byron Wallace. We also thank the ICML Debates organizers.

References

1. Armstrong, T.G., Moffat, A., Webber, W. and Zobel, J. Improvements that don't add up: ad-hoc retrieval results since 1998. In *Proceedings of the 18th ACM Conf. Information and Knowledge Management*, 2009, 601–610.
2. Bengio, Y. Practical recommendations for gradient-based training of deep architectures. *Neural Networks: Tricks of the Trade.* G. Montavon, G.B. Orr, KR Müller, eds. *LNCS* 7700 (2012). Springer, Berlin, Heidelberg, 437–78.
3. Bostrom, N. *Superintelligence.* Dunod, Paris, France, 2017.
4. Bottou, L. et al. Counterfactual reasoning and learning systems: The example of computational advertising. *J. Machine Learning Research 14*, 1 (2013), 3207–3260.
5. Bray, A.J. and Dean, D.S. Statistics of critical points of Gaussian fields on large-dimensional spaces. *Physical Review Letters 98*, 15 (2007), 150201; https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.98.150201.
6. Chen, D., Bolton, J. and Manning, C.D. A thorough examination of the CNN/*Daily Mail* reading comprehension task. In *Proceedings of the 54th Annual Meeting of Assoc. Computational Linguistics*, 2016, 2358–2367.
7. Choromanska, A., Henaff, M., Mathieu, M., Arous, G.B., LeCun, Y. The loss surfaces of multilayer networks. In *Proceedings of the 18th Intern. Conf. Artificial Intelligence and Statistics*, 2015.
8. Cohen, P.R., Howe, A.E. How evaluation guides AI research: the message still counts more than the medium. *AI Magazine 9*, 4 (1988), 35.
9. Cotterell, R., Mielke, S.J., Eisner, J. and Roark, B. Are all languages equally hard to language-model? In *Proceedings of Conf. North American Chapt. Assoc. Computational Linguistics: Human Language T echnologies*, Vol. 2, 2018.
10. Council of the European Union. Motion for a European Parliament Resolution with Recommendations to the Commission on Civil Law Rules on Robotics, 2016; https://bit.ly/285CBjM.
11. Dauphin, Y.N. et al. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in Neural Information Processing Systems*, 2014, 2933–2941.
12. Esteva, A. et al. Dermatologist-level classification of skin cancer with deep neural networks. *Nature 542*, 7639 (2017), 115-118.
13. Gershgorn, D. The data that transformed AI research—and possibly the world. Quartz, 2017; https://bit.ly/2uwyb8R.
14. Goodfellow, I.J., Vinyals, O. and Saxe, A.M. Qualitatively characterizing neural network optimization problems. In *Proceedings of the Intern. Conf. Learning Representations*, 2015.
15. Hazirbas, C., Leal-Taixé, L. and Cremers, D. Deep depth from focus. *arXiv Preprint*, 2017; *arXiv:1704.01085*.
16. He, K., Zhang, X., Ren, S. and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE Intern. Conf. Computer Vision*, 2015, 1026–1034.
17. Henderson, P. et al. Deep reinforcement learning that matters. In *Proceedings of the 32nd Assoc. Advancement of Artificial Intelligence Conf.*, 2018.
18. Ioffe, S. and Szegedy, C. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd Intern. Conf. Machine Learning 37*, 2015; http://proceedings.mlr.press/v37/ioffe15.pdf.
19. Kingma, D.P. and Ba, J. Adam: A method for stochastic optimization. In *Proceedings of the 3rd Intern. Conf. Learning Representations*, 2015
20. Knuth, D.E., Larrabee, T. and Roberts, P.M. Mathematical writing, 1987; https://bit.ly/2TmxyNq
21. Langley, P. and Kibler, D. The experimental study of machine learning, 1991; http://www.isle.org/~langley/papers/mlexp.ps.
22. Lipton, Z.C. The mythos of model interpretability. *Intern. Conf. Machine Learning Workshop on Human Interpretability*, 2016.
23. Lipton, Z.C., Chouldechova, A. and McAuley, J. Does mitigating ML's impact disparity require treatment disparity? *Advances in Neural Inform. Process. Syst.* 2017, 8136-8146. arXiv Preprint arXiv:1711.07076.
24. Lucic, M., Kurach, K., Michalski, M., Gelly, S., Bousquet, O. Are GANs created equal? A large-scale study. In *Proceedings of the 32nd Conf. Neural Information Processing Syst.* arXiv Preprint 2017; arXiv:1711.10337.
25. Markoff, J. Researchers announce advance in image-recognition software. *NYT* (Nov. 17, 2014); https://nyti.ms/2HfcmSe.
26. McDermott, D. Artificial intelligence meets natural stupidity. *ACM SIGART Bulletin 57* (1976), 4–9.
27. Melis, G., Dyer, C. and Blunsom, P. On the state of the art of evaluation in neural language models. In *Proceedings of the Intern. Conf. Learning Representations*, 2018.
28. Metz, C. You don't have to be Google to build an artificial brain. *Wired* (Sept. 26, 2014); https://www.wired.com/2014/09/google-artificial-brain/.
29. Minsky, M. *The Emotion Machine: Commonsense Thinking, Artificial Intelligence, and the Future of the Human Mind.* Simon & Schuster, New York, NY, 2006.
30. Mohamed, S., Lakshminarayanan, B. Learning in implicit generative models. arXiv Preprint, 2016; arXiv:1610.03483.
31. Noh, H., Hong, S. and Han, B. Learning deconvolution network for semantic segmentation. In *Proceedings of the Intern. Conf. Computer Vision*, 2015, 1520–1528.
32. Nye, M.J. N-rays: An episode in the history and psychology of science. *Historical Studies in the Physical Sciences 11*, 1 (1980), 125–56.
33. Open Science Collaboration. Estimating the reproducibility of psychological science. *Science 349*, 6251 (2015), aac4716.
34. Platt, J.R. Strong inference. *Science 146*, 3642 (1964), 347–353.
35. Reddi, S.J., Kale, S. and Kumar, S. On the convergence of Adam and beyond. In *Proceedings of the Intern. Conf. Learning Representations*, 2018.
36. Romer, P.M. Mathiness in the theory of economic growth. *Amer. Econ. Rev. 105*, 5 (2015), 89–93.
37. Santurkar, S., Tsipras, D., Ilyas, A. and Madry, A. How does batch normalization help optimization? (No, it is not about internal covariate shift). In *Proceedings of the 32nd Conf. Neural Information Processing Systems*; 2018; https://papers.nips.cc/paper/7515-how-does-batch-normalization-help-optimization.pdf.
38. Sculley, D., Snoek, J., Wiltschko, A. and Rahimi, A. Winner's curse? On pace, progress, and empirical rigor. In *Proceedings of the 6th Intern. Conf. Learning Representations*, Workshop Track, 2018
39. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Machine Learning Research 15*, 1 (2014), 1929–1958; https://dl.acm.org/citation.cfm?id=2670313.
40. Steinhardt, J. and Liang, P. Learning fast-mixing models for structured prediction. In *Proceedings of the 32nd Intern. Conf. Machine Learning 37* (2015), 1063–1072; http://proceedings.mlr.press/v37/steinhardtb15.html.
41. Steinhardt, J. and Liang, P. Reified context models. In *Proceedings of the 32nd Intern. Conf. Machine Learning 37*, (2015), 1043–1052; https://dl.acm.org/citation.cfm?id=3045230.
42. Steinhardt, J., Koh, P.W. and Liang, P.S. Certified defenses for data poisoning attacks. In *Proceedings of the 31st Conf. Neural Information Processing Systems*, 2017; https://papers.nips.cc/paper/6943-certified-defenses-for-data-poisoning-attacks.pdf.
43. Stock, P. and Cisse, M. ConvNets and ImageNet beyond accuracy: Explanations, bias detection, adversarial examples and model criticism. arXiv Preprint, 2017, arXiv:1711.11443.
44. Szegedy, C. et al. Intriguing properties of neural networks. *Intern. Conf. Learning Representations.* arXiv Preprint, 2013, arXiv:1312.6199.
45. Zellers, R., Yatskar, M., Thomson, S. and Choi, Y. Neural motifs: Scene graph parsing with global context. In *Proceedings of the IEEE Conf. Computer Vision and Pattern Recognition*, 2018, 5831–5840.
46. Zhang, C., Bengio, S., Hardt, M., Recht, B. and Vinyals, O. Understanding deep learning requires rethinking generalization. In *Proceedings of the Intern. Conf. Learning Representations*, 2017.

Zachary C. Lipton is an assistant professor at Carnegie Mellon University in the Tepper School of Business with appointments in the Machine Learning Department and the Heinz School of Public Policy. He also collaborates with Amazon, where he helped to grow AWS' Amazon AI team and contributed to the Apache MXNet deep learning framework. Find him at zacklipton.com, Twitter @zacharylipton, or GitHub @zackchase.

Jacob Steinhardt will be joining UC Berkeley as an assistant professor of statistics. He is a technical advisor for the Open Philanthropy Project and has collaborated with policy researchers to understand and avoid potential misuses of machine learning.

DOI:10.1145/3286588

Programmable software-defined solid-state drives can move computing functions closer to storage.

BY JAEYOUNG DO, SUDIPTA SENGUPTA, AND STEVEN SWANSON

Programmable Solid-State Storage in Future Cloud Datacenters

THERE IS A major disconnect today in cloud datacenters concerning the speed of innovation between application/operating system (OS) and storage infrastructures. Application/OS software is patched with new/improved functionality every few weeks at "cloud speed," while storage devices are off-limits for such sustained innovation during their hardware life cycle of three to five years in datacenters. Since the software inside the storage device is written by storage vendors as proprietary firmware not open for general application developers to modify, the developers are stuck with a device whose functionality and capabilities are frozen in time, even as many of them are modifiable in software. A period of five years is almost eternal in the cloud computing industry where new features, platforms, and application program interfaces (APIs) are evolving every couple of months and application-demanded requirements from the storage system grow quickly over time. This notable lag in the adaptability and velocity of movement of the storage infrastructure may ultimately affect the ability to innovate throughout the cloud world.

In this article, we advocate creating a software-defined storage substrate of solid-state drives (SSDs) that are as programmable, agile, and flexible as the applications/OS accessing from servers in cloud datacenters. A fully programmable storage substrate promises opportunities to better bridge the gap between application/OS needs and storage capabilities/limitations, while allowing application developers to innovate in-house at cloud speed.

The move toward software-defined control for IO devices and co-processors has played out before in the datacenter. Both GPUs and network interface cards (NICs) started as black-box devices that provide acceleration for CPU-intensive operations (such as graphics and packet processing). Internally, they implemented acceleration features with a combination of specialized hardware and proprietary firmware. As customers demanded greater flexibility, vendors slowly exposed programmability to the rest of the system, unleashing the vast processing power available from GPUs and a new level of agility in how systems can manage networks for enhanced functionality like more granular traffic management, security, and

» key insights

- A fully programmable storage substrate in cloud datacenters opens up new opportunities to innovate the storage infrastructure at cloud speed.
- In-storage programming is becoming increasingly easier with powerful processing capabilities and highly flexible development environments.
- New value propositions with the programmable storage substrate can be realized, such as customizing the storage interface, moving compute close to data, and performing secure computations.

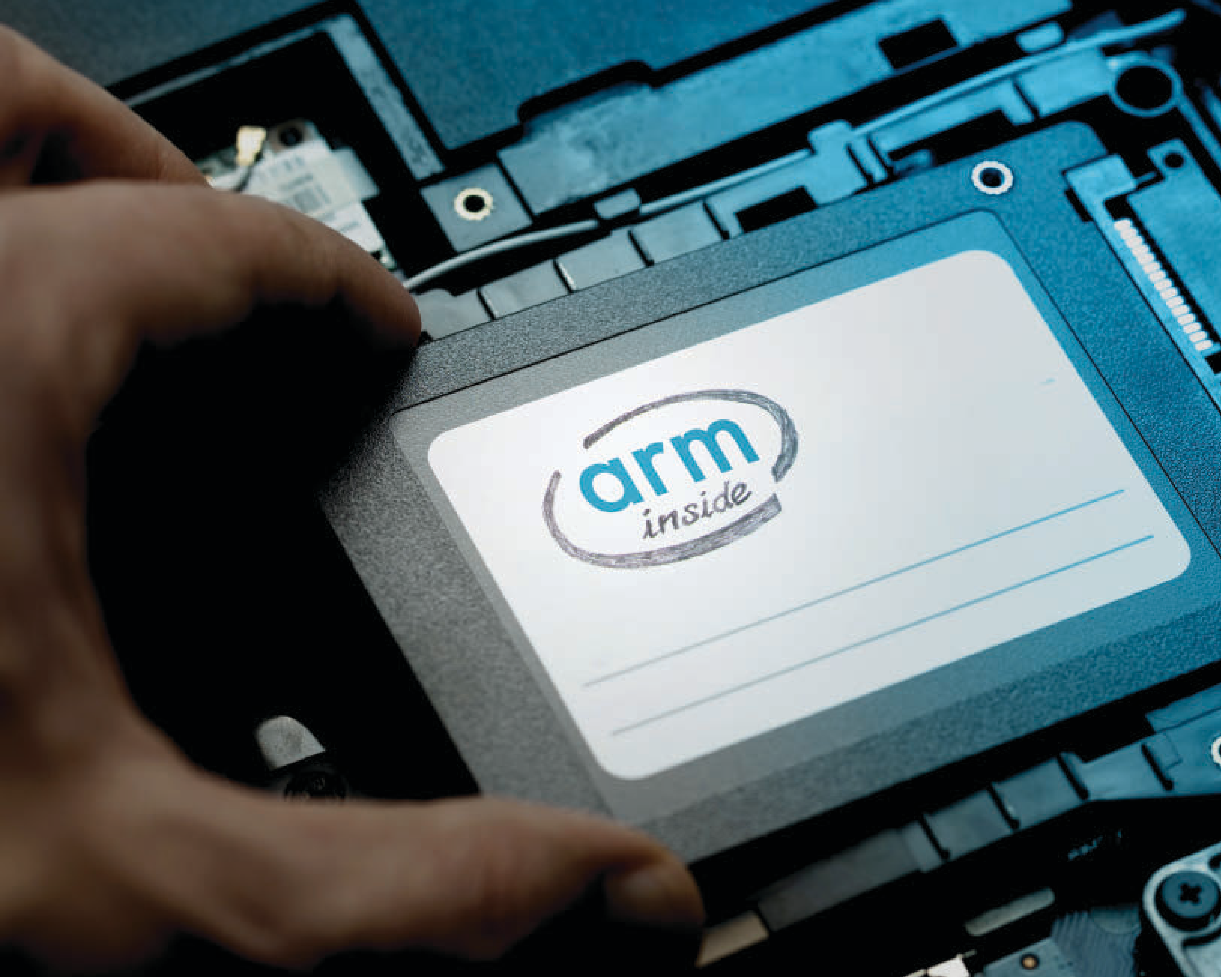


IMAGE BY ANDRIJ BORYS ASSOCIATES, USING PHOTO FROM SHUTTERSTOCK.COM

deep-network telemetry.

Storage is at the cusp of a similar transformation. Modern SSDs rely on sophisticated processing engines running complex firmware, and vendors already provide customized firmware builds for cloud operators. Exposing this programmability through easily accessible interfaces will let storage systems in the cloud datacenters adapt to rapidly changing requirements on the fly.

Storage Trends

The amount of data being generated daily is growing exponentially, placing more and more processing demand on datacenters. According to a 2017 marketing-trend report from IBM,[a] 90% of the data in the world in 2016 has been created in the last 12 months of 2015. Such large-scale datasets—which generally range from tens of terabytes to multiple petabytes—present challenges of extreme scale while achieving very fast and efficient data processing: a high-performance storage infrastructure in terms of throughput and latency is necessary. This trend has resulted in growing interest in the aggressive use of SSDs that, compared with traditional spinning hard disk drives (HDDs), provides orders-of-magnitude lower latency and higher throughput. In addition to these performance benefits, the advent of new technologies (such as 3D NAND enabling much denser chips and quad-level-cell, or QLC, for bulk storage) allows SSDs to continue to significantly scale in capacity and to yield a huge reduction in price.

There are two key components in SSDs,[4] as shown in Figure 1—an SSD controller and flash storage media. The controller that is most commonly implemented as a system-on-a-chip (SoC) is designed to manage the underlying storage media. For example, SSDs built using NAND flash memory have unique characteristics in that data can be written only to an empty memory location—no in-place updates are allowed—and memory can endure only a limited number of writes before it can no longer be read. Therefore, the controller must be able to perform some background management tasks (such as garbage collection) to reclaim flash blocks containing invalid data to create available space and wear leveling to evenly distribute writes across the entire flash blocks with the purpose of extending the SSD life. These tasks are, in general, implemented by proprietary firmware running on one or more embedded processor cores in

[a] https://ibm.co/2XNvHPk

Figure 1. Internal architecture of a modern flash SSD.

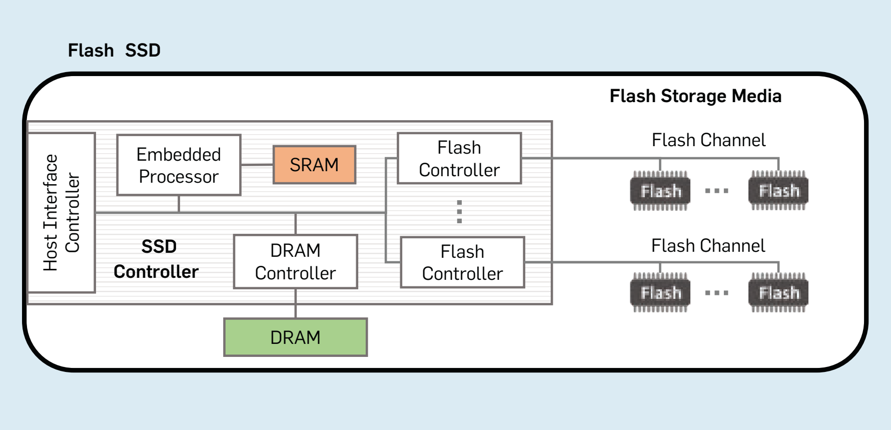
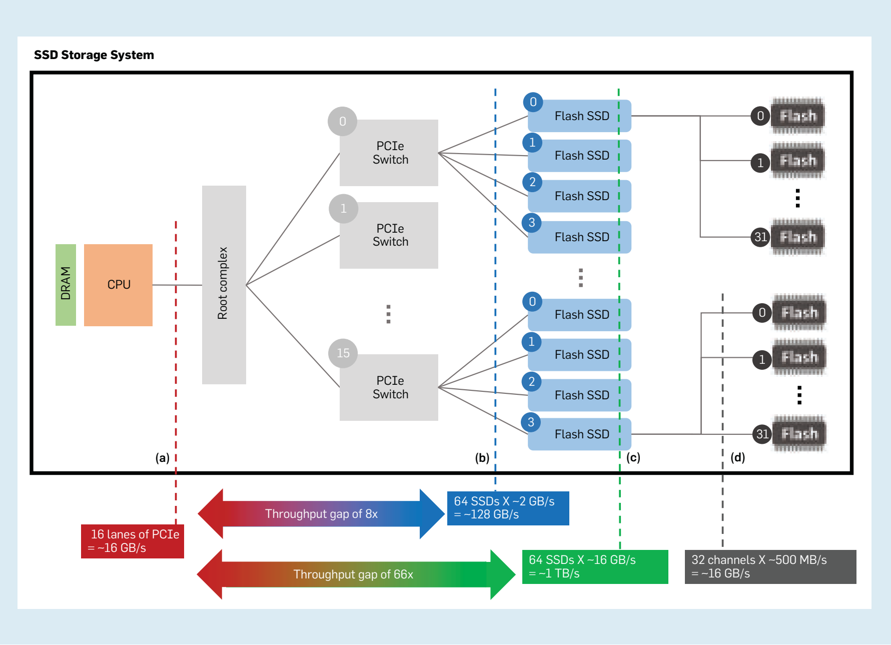


Figure 2. Example conventional storage server architecture with multiple NVMe SSDs.



the controller. In enterprise SSDs, large SRAM is often used for executing the SSD firmware, and both user data and internal SSD metadata are cached in external DRAM.

Interestingly, SSDs generally have a far larger aggregate internal bandwidth than the bandwidth supported by host I/O interfaces (such as SAS and PCIe). Figure 2 outlines an example of a conventional storage system that leverages a plurality of NVM Express (NVMe)[b] SSDs; 64 of them are connected to 16 PCIe switches that are mounted to a host machine via 16 lanes of PCIe Gen3. While this storage architecture provides a commodity solution for high-capacity storage server at low cost (compared to building a specialized server to directly attach all SSDs on the motherboard of the host), the maximum throughput is limited to 16-lane PCIe interface speed (see Figure 2a), which is approximately 16GB/sec, regardless of the number of SSDs accessed in parallel. There is thus an 8x throughput gap between the host interface and the total aggregated SSD bandwidth that could be up to roughly ~2GB/sec per SSD[c] X 64 SSDs = ~128GB/sec (see Figure 2b). More interestingly, this gap would grow further if the internal SSD performance is considered. A modern enterprise-level SSD usually consists of 16 or 32 flash channels, as outlined in Figure 2. Since each flash channel can keep up with ~500MB/sec; internally each SSD can be up to ~500MB/sec per channel X 32 channels = ~16GB/sec (see Figure 2d); and the total aggregated in-SSD performance would be ~16GB/sec per SSD X 64 SSDs = ~1TB/sec (see Figure 2c), a 66x gap. Making SSDs programmable would thus allow systems to fully leverage this abundant bandwidth.

In-Storage Programming

Modern SSDs combine processing—embedded processor—and storage components—SRAM, DRAM, and flash memory—to carry out routine functions required for managing the SSD. These computing resources present interesting opportunities to run general user-defined programs. In 2013, Do et al.[6,17] explored such opportunities for the first time in the context of running selected database operations inside a Samsung SAS flash SSD. They wrote simple selection and aggregation operators that were compiled into the SSD firmware and extended the execution framework of Microsoft SQL Server 2012 to develop a working prototype in which simple selection and aggregation queries could be run end-to-end.

That work demonstrated several times improvement in performance and energy efficiency by offloading database operations onto the SSD and highlighted a number of challenges that would need to be overcome to broadly adapt programmable SSDs: First, the computing capabilities available inside the SSD are limited by design. The low-performance embedded processor inside the SSD without L1/L2 caches and high latency to the in-SSD DRAM require extra careful programming to run user code in the SSD without producing a performance bottleneck.

Moreover, the embedded software-development process is complex and makes programming and debugging very challenging. To maximize performance, Do et al. had to carefully plan the layout of data structures used by the code running inside the SSD to avoid spilling out of the SRAM. Likewise, Do et al. used a hardware-debugging tool to debug programs running inside the SSD that is far more primitive than reg-

[b] A device interface for accessing non-volatile memory attached via a PCI Express (PCIe) bus.

[c] Practical sequential-read bandwidth of a commodity PCIe SSD.

ular debugging tools (such as Microsoft Visual Studio) available to general application developers. Worse, the device-side processing code—selection and aggregation—had to be compiled into the SSD firmware in the prototype, meaning application developers would need to worry about not only the target application itself but also complex internal structures and algorithms in the SSD firmware.

On top of this, the consequences of an error can be quite severe, which could result in corrupted data or an unusable drive. Workaday application programmers are unlikely to accept the additional complexity, and cloud providers are unlikely to let untrusted code run in such a fragile environment.

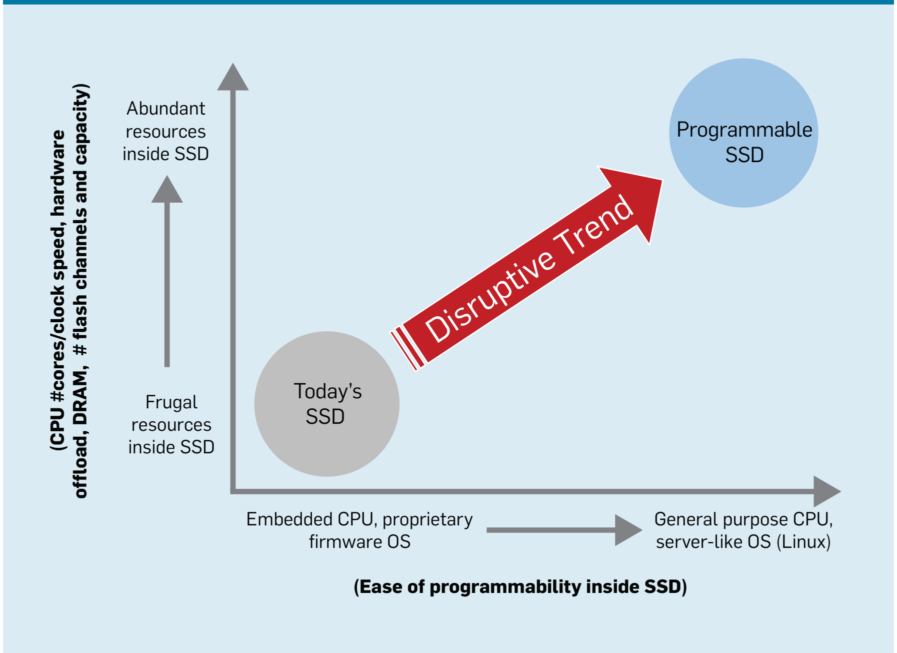
Application developers need a flexible and general programming model that allows easily running user code written in a high-level programming language (such as C/C++) inside an SSD. The programming model must also support the concurrent execution of multiple in-SSD applications while ensuring that malicious applications do not adversely affect the overall SSD operation or violate protection guarantees provided by the operating and file system.

In 2014, Seshadri et al.[20] proposed Willow, an SSD that made programmability a central feature of the SSD interface, allowing ordinary developers to safely augment and extend the SSD semantics with application-specific functions without compromising file system protections. In their model, host and in-SSD applications communicate via PCIe using a simple, generic—not storage-centric—remote procedure call (RPC) mechanism. In 2016, Gu et al.[7] explored a flow-based programming model where an in-SSD application can be constructed from tasks and data pipes connecting the tasks. These programming models provide great flexibility in terms of programmability but are still far from "general purpose." There is a risk that existing large applications might still need significant redesigns to exploit each model's capabilities, requiring much time and effort.

Fortunately, winds of change can disrupt the industry and help application developers explore SSD programming in a better way, as illustrated in Figure 3. The processing capabilities available inside the SSD are increasingly powerful, with abundant compute and bandwidth resources. Emerging SSDs include software-programmable controllers with multi-core processors, built-in hardware accelerators to offload compute-intensive tasks from the processors, multiple GBs of DRAM, and tens of independent channels to the underlying storage media, allowing several GB/s of internal data throughput. Even more interesting and useful, programming SSDs is becoming easier, with the trend away from proprietary architectures and software runtimes and toward commodity operating systems (such as Linux) running on top of general-purpose processors (such as ARM and RISC-V). This trend enables general application developers to fully leverage existing tools, libraries, and expertise, allowing them to focus on their own core competencies rather than spending many hours getting used to the low-level, embedded development process. This also allows application developers to easily port large applications already running on host operating systems to the device with minimal code changes.

All in all, the programmability evolution in SSDs presents a unique opportunity to embrace the SSDs as a first-class programmable platform in the cloud datacenters, enabling software-hardware innovation inside the SSD. Moreover, going beyond the packaged SSD, because the two major components inside the SSD are each manufactured by multiple vendors,[d] it is conceivable that SSDs could be custom designed and provided in partnership with component vendors[e] (just like how today's datacenter servers are built and deployed), and even contribute back some of the designs to the community (via forums like the Open Compute project, https://www.opencompute.org). For example, the industry is already moving in this direction with introduction of the Open-Channel SSD technology[2,8,f] that moves much of the SSD firmware functionalities out of the black box and into the operating system or userspace, giving applications better control over the device. In an open source project called Denali[g] in 2018, Microsoft proposed a scheme

Figure 3. Disruptive trends in the flash storage industry toward abundant resources and increased ease of programmability inside the SSD.


(CPU #cores/clock speed, hardware offload, DRAM, # flash channels and capacity)
Abundant resources inside SSD
Frugal resources inside SSD
Programmable SSD
Disruptive Trend
Today's SSD
Embedded CPU, proprietary firmware OS
General purpose CPU, server-like OS (Linux)
(Ease of programmability inside SSD)

[d] Several vendors manufacture each type of component in flash SSDs. For example: flash controller manufactured by Marvell, PMC (acquired by Microsemi), Sandforce (acquired by Seagate), Indilinx (acquired by OCZ), and flash memory manufactured by Samsung, Toshiba, and Micron.

[e] Many large-scale datacenter operators (such as Google[19] and Baidu[16]) build their own SSDs that are fully optimized for their own application requirements.

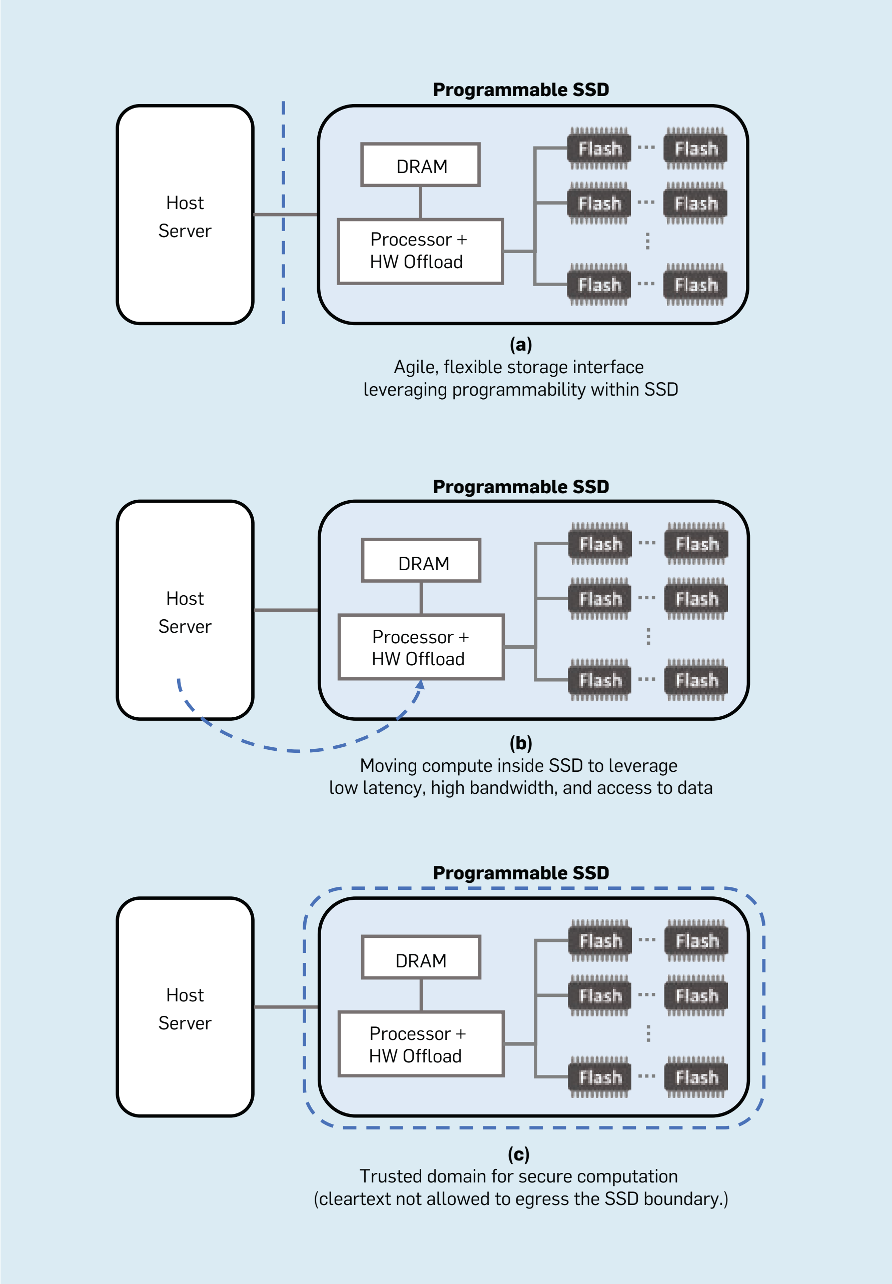
[f] The Linux Open-Channel SSD subsystem was introduced in the Linux kernel version 4.4.

[g] https://bit.ly/2GCuIum

that splits the monolithic components of an SSD into two different modules—one standardized part dealing with storage media and a software interface to handle application-specific tasks (such as garbage collection and wear leveling). In this way, SSD suppliers can build simpler products for datacenters and deliver them to market more quickly while per-application tuning is possible by datacenter operators.

The component-based ecosystem also opens up entirely new opportunities for integrating powerful heterogeneous programming elements (such as field-programmable gate array (FPGA)[13,21] and GPU,[h] with storage media) and flash and other emerging new non-volatile memories (such as 3D XPoint, ReRAM, STT-RAM, and PCM) that provide persistent storage at DRAM latencies to deliver high-performance gains. This approach would present the greatest flexibility to take advantage of advances in the underlying storage device to optimize performance for multiple cloud applications. In the near future, the software-hardware innovation inside the SSD can proceed much like the PC, networking hardware, and GPU ecosystems have in the past. This is an opportunity to rethink datacenter architecture with efficient use of heterogeneous, energy-efficient hardware, which is the way forward for higher performance at lower power.

Figure 4. Programmable SSD value proposition.



(a)
Agile, flexible storage interface leveraging programmability within SSD

(b)
Moving compute inside SSD to leverage low latency, high bandwidth, and access to data

(c)
Trusted domain for secure computation (cleartext not allowed to egress the SSD boundary.)

Value Propositions

Here, we summarize three value propositions that demonstrate future directions in programmable storage (see Figure 4):

Agile, flexible storage interface (see Figure 4a). Full programmability will allow the storage interface and feature set to evolve at cloud speed, without having to persuade standardization bodies to bless them or persuade device manufacturers to implement them in the next-generation hardware roadmap, both usually involving years of delay. A richer, customizable storage interface will allow application developers to stay focused on their application, without having to work around storage constraints, quirks, or peculiarities, thus improving developer productivity.

As an example of the need for such an interface, consider how stream writes are handled in the SSD today. Because the SSD cannot differentiate between incoming data from multiple streams, it could pack data from different streams onto the same flash erase block, the smallest unit that can be erased from flash at once. When a portion of the stream data is deleted, it leaves blocks with holes of invalid data. To reclaim these blocks, the garbage-collection activity inside the SSD must copy around the valid data, slowing the device and increasing write amplification, thus reducing device lifetime.

If application developers had control over the software inside the SSD, they could handle streams much more efficiently. For instance, incoming writes could be tagged with stream IDs and the device could use this information to fill a block with data from the same stream. When data from that stream is deleted, the entire data block could be reclaimed without copying around data. Such stream awareness has been shown to double device lifetime, significantly increasing read performance.[14] In Microsoft, this need of supporting multiple streams in the SSD was identified in 2014, but NVMe incorporated the fea-

[h] https://bit.ly/2L8LfM4

ture only late 2017.[i] Moreover, large-scale deployment in Microsoft datacenters might take at least another year and be very expensive, since new SSDs must be purchased to essentially get a new version of the firmware. Waiting five years for a change to a system software component is completely out of step with how quickly computer systems are evolving today. A programmable storage platform would reduce this delay to months and allow rapid iteration and refinement of the feature, not to mention the ability to "tweak" the implementation to match specific use cases.

Moving compute close to data (see Figure 4b). The need to analyze and glean intelligence from big data imposes a shift from the traditional compute-centric model to data-centric model. In many big data scenarios, application performance and responsiveness (demanded by interactive usage) is dominated not by the execution of arithmetic and logic instructions but instead by the requirement to handle huge volumes of data and the cost of moving this data to the location(s) where compute is performed. When this is the case, moving the compute closer to the data can reap huge benefits in terms of increased throughput, lower latency, and reduced energy usage.

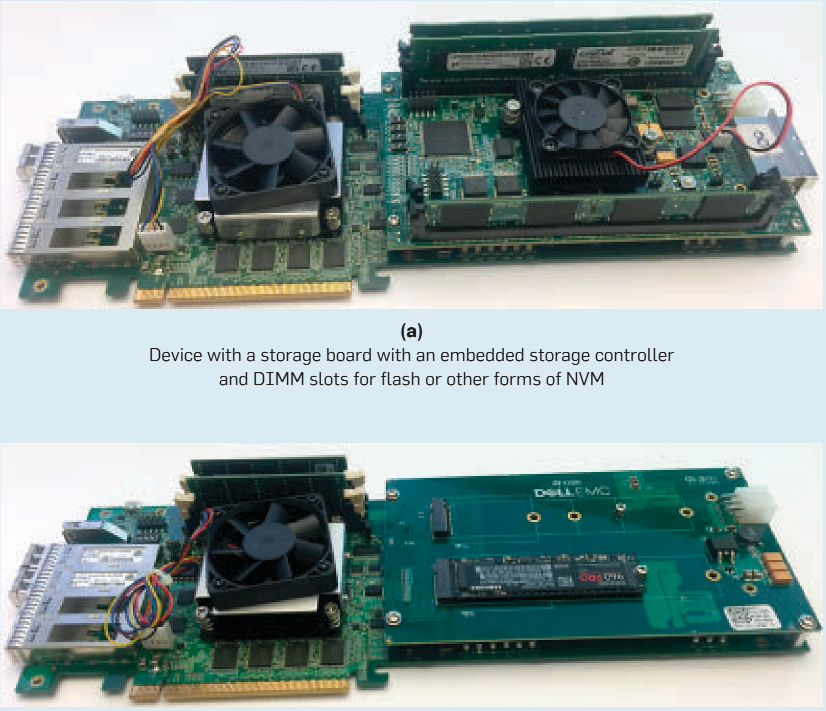
Big data analytics running inside an SSD can have access to the stored data with tens of GB/sec bandwidth (rivaling DRAM bandwidth), and with latency comparable to accessing raw non-volatile memory. In addition, large energy savings can be achieved because processors inside the SSD are more energy efficient compared to the host-server CPU (such as Intel Xeon), and data does not need to be hauled over large distances from storage all the way up to the host via network, which is more energy-expensive than processing it.

Processors inside the SSD are clearly not as powerful as host processors, but together with in-storage hardware offload engines, a broad range of data processing tasks can be competitively performed inside the SSD. As an example, consider how data analytic queries are processed in general: When an analytic query is given, compressed data required to answer the query is first loaded to host, uncompressed, and then executed using host resources. Such fundamental data analytics primitive can be processed inside the SSD by accessing data with high internal bandwidth and by offloading decompression to the dedicated engine. Subsequent stages of the query-processing pipeline (such as filtering out unnecessary data and performing the aggregation) can execute inside the SSD, resulting in greatly reduced network traffic and saved host CPU/memory resources for other important jobs. Further, performance and bandwidth together can be scaled by adding more SSDs to the system if the application requires higher data rates.

Secure computation in the cloud (see Figure 4c). Recent security breach events related to personal, private information (financial and otherwise) have exposed the vulnerability of data infrastructures to hackers and attackers. Also, a new type of malicious software called "encryption ransomware" attacks machines by stealthily encrypting data files and demanding a ransom to provide access to these files.[10] Security is often among the topmost concerns enterprise chief information officers have when they move to the cloud, as cloud providers are unwilling to take on full liability for the impact of such breaches. Development of a secure cloud is not just a feature requirement but also an absolute foundational capability necessary for the future of the cloud computing model and its business success as an industry.

To realize the vision of a trusted cloud, data must be encrypted while stored at rest, which however, limits the kind of computation that can be performed on encrypted data without decryption. To facilitate arbitrary (legitimate) computation on stored data, it needs to be decrypted before computing on it. This requires decrypted cleartext data to be present (at least temporarily) in various portions of the datacenter infrastructure vulnerable to security attacks. Application developers need a way to facilitate secure computation on the cloud by fencing in well-defined, narrow, trusted domains that can preserve the ability to perform arbitrary com-

Figure 5. A prototype programmable SSD developed for research purposes.



(a)
Device with a storage board with an embedded storage controller
and DIMM slots for flash or other forms of NVM

(b)
Device with a storage board where M.2 SSDs can be plugged into.

[i] Note the multi-stream technology for SCSI/SAS was standardized in T10 on May 20, 2015.

putation on the data.

SSDs with their powerful compute capabilities can form a trusted domain for doing secure computation on encrypted data, leveraging their internal hardware cryptographic engine and secure boot mechanisms for this purpose. Cryptographic keys can be stored inside the SSD, allowing arbitrary compute to be carried out on the stored data—after decryption if needed—while enforcing that data cannot leave the device in cleartext form. This allows a new, flexible, easily programmable, near-data realization of trusted hardware in the cloud. Compared to currently proposed solutions like Intel Enclaves[j] that are protected, isolated areas of execution in the host server memory, this solution protects orders of magnitude more data.

Programmable SSDs

While the concept of in-storage processing on SSDs was proposed more than six years ago,[6] experimenting with SSD programming has been limited by the availability of real hardware on which a prototype can be built to demonstrate what is possible. The recent emergence of prototyping boards available for both research and commercial purposes has opened new opportunities for application developers to take ideas from conception to action.

Figure 5 shows such prototype device, called Dragon Fire Card (DFC),[k,3,5] designed and manufactured by Dell EMC and NXP for research. The card is powerful and flexible with enterprise-level capabilities and resources. It comprises a main board and a storage board. The main board contains an ARMv8 processor, 16GB of RAM, and various on-chip hardware accelerators (such as 20Gbps compression/decompression, 20Gbps SEC-crypto, and 10Gbps RegEx engines). It also provides NVMe connectivity via four PCIe Gen3 lanes, and 4x10Gbps Ethernet that supports remote dynamic memory access (RDMA) over converged ethernet (RoCE) protocol. It supports two different storage boards that connect via 2x4 PCIe Gen3 lanes: One type of board (see Figure 5a) includes an embedded storage controller and four memory slots where flash or other forms of NVM can be installed; and the second (see Figure 5b) an adapter that hosts two M.2 SSDs.

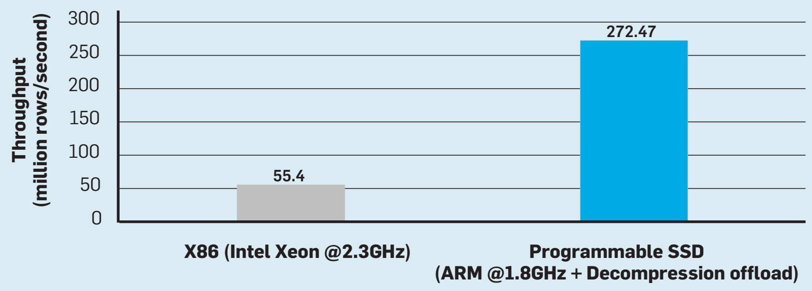
The ARM SoC inside the board runs a full-fledged Ubuntu Linux, so programming the board is very similar to programming any other Linux device. For instance, software can leverage the Linux container technology (such as Docker) to provide isolated environments inside the board. To create applications running on the board, a software development kit (SDK) containing GNU tools to build applications for ARM and user/kernel mode libraries to use the on-chip hardware accelerators is provided, allowing a high level of programmability. The DFC can also serve as a block device, just like regular SSDs. For this purpose, the device is shipped with a flash translation layer (FTL) that runs on the main board.

The SSD industry is also moving toward bringing compute to SSDs so data can be processed without leaving the place where it is originally stored. For instance, in 2017 NGD Systems[l] announced an SSD called Catalina2[1] capable of running applications directly on the device. Catalina2 uses TLC 3D NAND flash (up to 24TB), which is connected to the onboard ARM SoC that runs an embedded Linux and modules for error-correcting code (ECC) and FTL. On the host server, a tunnel agent (with C/C++ libraries) runs to talk to the device through the NVMe protocol. As another example, ScaleFlux[m] uses a Xilinx FPGA (combined with terabytes of TLC 3D NAND flash) to compute data for data-intensive applications. The host server runs a software module, providing API accesses to the device while being responsible for FTL and flash-management functionalities.

Academia and industry are working to establish a compelling value proposition by demonstrating application scenarios for each of the three pillars outlined in Figure 4. Among them we are initially focused on exploring the benefits and challenges of moving compute closer to storage (see Figure 4b) in the context of big data analytics, examining large amounts of data to uncover hidden patterns and insights.

Big data analytics within a programmable SSD. To demonstrate our approach, we have implemented a C++ reader that runs on a DFC card (see Figure 5) for Apache Optimized Row Columnar (ORC) files. The ORC file format is designed for fast processing and high storage efficiency of big data analytic workloads, and has been widely adopted in the open source community and industry. The reader running inside the SSD reads large chunks of ORC streams, decompresses them, and then evaluates query predicates to find only necessary values. Due to the server-like development environment—Ubuntu and a general-purpose ARM processor—we easily ported a reference implementation of the ORC reader[n] to the ARM SoC environment (with only a few lines of code changes) and incorporat-

Figure 6. Preliminary results using a programmable SSD yield approximately 5x speedups for full scans of ZLIB-compressed ORC files within the device, compared to native ORC readers running on x86 architecture.



[j] https://software.intel.com/en-us/sgx

[k] https://github.com/DFC-OpenSource

[l] http://www.ngdsystems.com

[m] http://www.scaleflux.com

[n] https://github.com/apache/orc

ed library APIs into the reader, enabling reading data from flash and offloading the decompression work to the ARM SoC hardware accelerator.

Figure 6 shows preliminary bandwidth results of scanning a ZLIB-compressed, single-column integer dataset (one billion rows) through the C++ ORC reader running on a host x86 server vs. inside the DFC card, respectively.[o] As in the figure, we achieved approximately 5x faster scan performance inside the device compared to running on the host server. Given that this is a single device performance, we should be able to achieve much better performance improvements by increasing the number of programmable SSDs that are used in parallel.

In addition to scanning, filtering, and aggregating large volumes of data at high-throughput rates by offloading part of the computation directly to the storage has been explored as well. In 2016 Jo et al.[12] built a prototype that performs very early filtering of data through a combination of ARM and a hardware pattern-matching engine available inside a programmable SSD equipped with a flow-based programming model described by Gu et al.[7] When a query is given, the query planner determines whether early filtering is beneficial for the query and chooses a candidate table as the target if the estimated filtering ratio is sufficiently high. Early filtering is then performed against the target table inside the device, and only filtered data is then fetched to the host for residual computation. This early filtering inside the device turns out to be highly effective for analytic queries; when running all 22 TPC-H queries on a MariaDB server with the programmable device prototyped on a commodity NVMe SSD, a 3.6x speedup was achieved by Jo et al.[12] compared to a system with the same SSD without the programmability.

Alternatively, an FPGA-based prototype design for near-data processing inside the a storage node for database engines was studied by István et al.[11] in 2017. In this prototype, each storage node that could be accessed over the network through a simple key-value store interface provided fault tolerance through replication and application-specific processing (such as predicate evaluations, substring matching and decompression) at line rate.

The programmable storage substrate can be viewed as a hyper-converged infrastructure where storage, networking, and compute are tightly coupled for low-latency, high-throughput access, while still providing availability.

Datacenter Realization

Each application running in cloud datacenters has its own, unique requirements, making it difficult to design server nodes with the proper balance of compute, memory, and storage. To cope with such complexity, an approach of physically decoupling resources was proposed recently by Han et al.[9] in 2013 to allow replacing, upgrading, or adding individual resources instead of the entire node. With the availability of fast interconnect technologies (such as InfiniBand, RDMA, and RoCE), it is already common in today's large-scale cloud datacenters to disaggregate storage from compute, significantly reducing the total cost of ownership and improving the efficiency of the storage utilization. However, storage disaggregation is a challenge[15] as storage-media access latencies are heading toward single-digit microsecond level[p] compared to a disk's millisecond latency, which is much larger than the fast network overhead. It is likely that, in the next few years the network latency will become a bottleneck as new, emerging non-volatile memories with extremely low latencies become available.

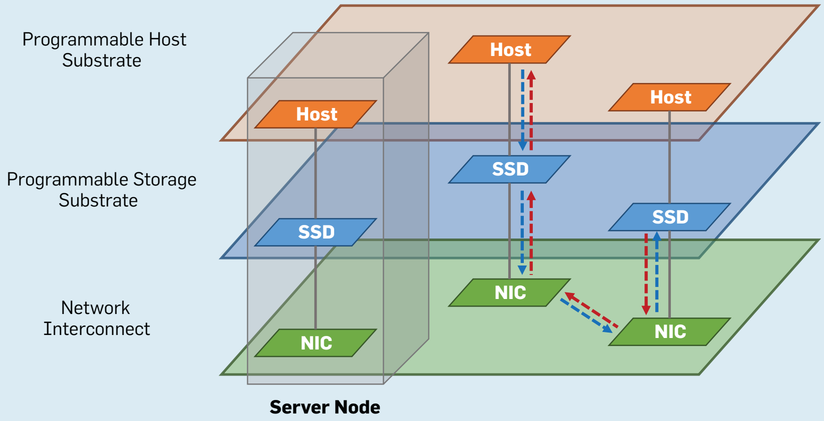
This challenge of storage disaggregation can be overcome by using programmable storage, enabling a fully programmable storage substrate that is decoupled from the host substrate as outlined in Figure 7. This view of storage as a programmable substrate allows application developers not only to leverage very low, storage-medium access latency by running programs inside the storage device but also to access any remote storage device without involving the remote host server where the device is physically attached (see Figure 7) by

[o] Note, to effectively compare data-processing capability in each case—Intel Xeon in x86 vs. ARM + decompression accelerator in the device—only a single core for each processor was used.

[p] For example, the access latency of 3D XPoint can take 5˜10 µsec, while NVMe SSD and disk takes ˜50–100 µsec and 10 msec, respectively.[8]

Figure 7. Enabling a programmable storage substrate decoupled from the host substrate.

Direct traffic between programmable storage devices (with a network interface) without involving a remote host.



using NVMe over Fabrics (NVMe-oF)[q] with RDMA.

With the programmable storage substrate, we can think of going beyond the single-device block interface. For example, a micro server inside storage can expose a richer interface like a distributed key-value store or distributed streams. Or the storage infrastructure can be managed as a fabric, not as individual devices. The programmable storage substrate can also provide high-level datacenter capabilities (such as backup, data snapshot, replication, de-duplications, and tiering), which are typically supported in a datacenter server environment where compute and storage are separated. This means the programmable storage substrate can be viewed as a hyper-converged infrastructure where storage, networking, and compute are tightly coupled for low-latency, high-throughput access, while still providing availability.

Conclusion

In this article, we have presented our vision of a fully programmable storage substrate in cloud datacenters, allowing application developers to innovate the storage infrastructure at cloud speed like the software application/OS infrastructure. The programmability evolution in SSDs provides opportunities for embracing them as a first-class programmable platform in cloud datacenters, enabling software-hardware innovation that could bridge the gap between application/OS needs and storage capabilities/limitations. We hope to shed light on the future of software-defined storage and help chart a direction for designing, building, deploying, and leveraging a software-defined storage architecture for cloud datacenters.

Acknowledgments

This work was supported in part by National Science Foundation Award 1629395.

[q] A technology specification designed for non-volatile memories to transfer data between a host and a target system/device over a network. Approximately 90% of the NVMe-oF protocol is the same as the NVMe protocol.

References

1. Alves, V. In-situ processing. Flash Memory Summit (Santa Clara, CA, Aug. 8–10), 2017.
2. Bjørling, M., González, J., and Bonnet, P. Lightnvm: The Linux open-channel SSD subsystem. In *Proceedings of the 15th USENIX Conference on File and Storage Technologies* (Santa Clara, CA, Feb. 27–Mar. 2). USENIX Association, Berkeley, CA, 2017, 359–374.
3. Bonnet, P. What's up with the storage hierarchy? In *Proceedings of the 8th Biennial Conference on Innovative Data Systems Research* (Chaminade, CA, Jan. 8–11), 2017.
4. Cornwell, M. Anatomy of a solid-state drive. *Commun. ACM 55*, 12 (Dec. 2012), 59–63.
5. Do, J. Softflash: Programmable storage in future data centers. In *Proceedings of the 20th SNIA Storage Developer Conference* (Santa Clara, CA, Sep. 11–14), 2017.
6. Do, J., Kee, Y.-S., Patel, J.M., Park, C., Park, K., and DeWitt, D.J. Query processing on smart SSDs: Opportunities and challenges. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (New York, NY, Jun. 22–27). ACM Press, New York, 2013, 1221–1230.
7. Gu, B., Yoon, A. S., Bae, D.-H., Jo, I., Lee, J., Yoon, J., Kang, J.-U., Kwon, M., Yoon, C., Cho, S., et al. Biscuit: A framework for near data processing of big data workloads. In *Proceedings of the ACM/IEEE 43rd Annual International Symposium on Computer Architecture* (Seoul, S. Korea, Jun. 18–22). IEEE, 2016, 153-165.
8. Hady, F. Wicked fast storage and beyond. In *Proceedings of the 7th Non Volatile Memory Workshop* (San Diego, CA, Mar. 6–8). Keynote, 2016.
9. Han, S., Egi, N., Panda, A., Ratnasamy, S., Shi, G., and Shenker, S. Network support for resource disaggregation in next-generation datacenters. In *Proceedings of the 12th ACM Workshop on Hot Topics in Networks* (College Park, MD, Nov. 21–22). ACM Press, New York, 2013, 10.
10. Huang, J., Xu, J., Xing, X., Liu, P., and Qureshi, M. K. Flashguard: Leveraging intrinsic flash properties to defend against encryption ransomware. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, TX, Oct. 30–Nov. 3). ACM Press, New York, 2017, 2231–2244.
11. István, Z., Sidler, D., and Alonso, G. Caribou: Intelligent distributed storage. In *Proceedings of the VLDB Endowment 10*, 11 (Aug. 2017), 1202–1213.
12. Jo, I., Bae, D.-H., Yoon, A.S., Kang, J.-U., Cho, S., Lee, D.D., and Jeong, J. YourSQL: A high-performance database system leveraging in storage computing. In *Proceedings of the VLDB Endowment* 9, 12 (Aug. 2016), 924–935.
13. Jun, S.-W., Liu, M., Lee, S., Hicks, J., Ankcorn, J., King, M., Xu, S., et al. BlueDBM: An appliance for big data analytics. In *Proceedings of the ACM/IEEE 42nd Annual International Symposium on Computer Architecture* (Portland, OR, Jun. 13–17). IEEE, 2015, 1–13.
14. Kang, J.-U., Hyun, J., Maeng, H., and Cho, S. The multi-streamed solid-state drive. In *Proceedings of the 6th USENIX Workshop on Hot Topics in Storage and File Systems* (Philadelphia, PA, Jun. 17–18). USENIX Association, Berkeley, CA, 2014.
15. Klimovic, A., Kozyrakis, C., Thereska, E., John, B., and Kumar, S. Flash storage disaggregation. In *Proceedings of the 11th European Conference on Computer Systems* (London, U.K., Apr. 18–21). ACM Press, New York, 2016, 29.
16. Ouyang, J., Lin, S., Jiang, S., Hou, Z., Wang, Y., and Wang, Y. SDF: Software-defined flash for web-scale Internet storage systems. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems* (Salt Lake City, UT, Mar. 1–5). ACM press, New York, 2014, 471–484.
17. Park, K., Kee, Y.-S., Patel, J.M., Do, J., Park, C., and Dewitt, D.J. Query processing on smart SSDs. *IEEE Data Engineering Bulletin 37*, 2 (Jun. 2014), 19–26.
18. Picoli, I.L., Pasco, C.V., Jónsson, B.Þ., Bouganim, L., and Bonnet, P. uFLIP-OC: Understanding flash I/O patterns on open-channel solid state drives. In *Proceedings of the 8th Asia-Pacific Workshop on Systems* (Mumbai, India, Sep. 2–3). ACM Press, New York, 2017, 20.
19. Schroeder, B., Lagisetty, R., and Merchant, A. Flash reliability in production: The expected and the unexpected. In *Proceedings of the 14th USENIX Conference on File and Storage Technologies* (Santa Clara, CA, Feb. 22–25). USENIX Association, Berkeley, CA, 2016, 67–80.
20. Seshadri, S., Gahagan, M., Bhaskaran, S., Bunker, T., De, A., Jin, Y., Liu, Y., and Swanson, S. Willow: A user-programmable SSD. In *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation* (Broomfield, CO, Oct. 6–8). USENIX Association, Berkeley, CA, 2014, 67–80.
21. Woods, L., István , Z., and Alonso, G. Ibex: An intelligent storage engine with support for advanced SQL offloading. In *Proceedings of the VLDB Endowment 7*, 11 (Jul. 2014), 963–974.

Jaeyoung Do (jaedo@microsoft.com) is a researcher at Microsoft Research, Redmond, WA, USA. He is leading a project, SoftFlash, which aims to use programmable SSDs in cloud datacenters.

Sudipta Sengupta (sudipta@amazon.com) is leading new initiatives in artificial intelligence/deep learning at Amazon AWS, Seattle, WA, USA; the research reported in this article was done while he was at Microsoft Research, Redmond, WA, USA.

Steven Swanson (swanson@cs.ucsd.edu) is a professor in the Department of Computer Science and Engineering at the University of California, San Diego, USA.

DOI:10.1145/3282487

Cybersecurity design reduces the risk of system failure from cyberattack, aiming to maximize mission effectiveness.

BY O. SAMI SAYDJARI

Engineering Trustworthy Systems: A Principled Approach to Cybersecurity

CYBERATTACKS ARE INCREASING in frequency, severity, and sophistication. Target systems are becoming increasingly complex with a multitude of subtle dependencies. Designs and implementations continue to exhibit flaws that could be avoided with well-known computer-science and engineering techniques. Cybersecurity technology is advancing, but too slowly to keep pace with the threat. In short, cybersecurity is losing the escalation battle with cyberattack. The results include mounting damages in the hundreds of billions of dollars,[4] erosion of trust in conducting business and collaboration in cyberspace, and risk of a series of catastrophic events that could cause crippling damage to companies and even entire countries. Cyberspace is unsafe and is becoming less safe every day.

The cybersecurity discipline has created useful technology against aspects of the expansive space of possible cyberattacks. Through many real-life engagements between cyberattackers and defenders, both sides have learned a great deal about how to

» key insights

- Cybersecurity must be practiced as a principled engineering discipline.
- Many principles derive from insight into the nature of how cyberattacks succeed.
- Defense in depth and breath is required to cover the spectrum of cyberattack classes.

design attacks and defenses. It is now time to begin abstracting and codifying this knowledge into principles of cybersecurity engineering. Such principles offer an opportunity to multiply the effectiveness of existing technology and mature the discipline so that new knowledge has a solid foundation on which to build.

Engineering Trustworthy Systems[8] contains 223 principles organized into 25 chapters. This article will address 10 of the most fundamental principles that span several important categories and will offer rationale and some guidance on application of those principles to design. Under each primary principle, related principles are also included as part of the discussion.

For those so inclined to read more in *Engineering Trustworthy Systems*, after each stated principle is a reference of the form "{x.y}" where x is the chapter number in which it appears and y is the y-th principle listed in that chapter (which are not explicitly numbered in the book).

Motivation

Society has reached a point where it is inexorably dependent on trustworthy systems. Just-in-time manufacturing, while achieving great efficiencies, creates great fragility to cyberattack, amplifying risk by allowing effects to propagate to multiple systems {01.06}. This means that the potential harm from a cyberattack is increasing and now poses existential threat to institutions. Cybersecurity is no longer the exclusive realm of the geeks and nerds, but now must be considered as an essential risk to manage alongside other major risks to the existence of those institutions.

The need for trustworthy systems extends well beyond pure technology. Virtually everything is a system from some perspective. In particular, essential societal functions such as the military, law enforcement, courts, societal safety nets, and the election process are all systems. People and their beliefs are systems and form a component of larger societal systems, such as voting. In 2016, the world saw cyberattacks transcend technology targets to that of wetware—human beliefs and propensity to action. The notion of hacking democracy itself came into light,[10] posing an existential threat to entire governments and ways of life though what is sometimes known by the military as *influence operations*{24.09}.[6]

Students of cybersecurity must be students of cyberattacks and adversarial behavior.

Before launching into the principles, one more important point needs to be made: *Engineers are responsible for the safety and security of the systems they build* {19.13}. In a conversation with my mentor's mentor, I once made the mistake of using the word *customer* to refer to those using the cybersecurity systems we were designing. I will always remember him sharply cutting me off and telling me that they were "clients, not customers." He said, "Used-car salesmen have customers; we have clients." Like doctors and lawyers, engineers have a solemn and high moral responsibility to do the right thing and keep those who use our systems safe from harm to the maximum extent possible, while informing them of the risks they take when using our systems.

In *The Thin Book of Naming Elephants*,[5] the authors describe how the National Aeronautics and Space Administration (NASA) shuttle-engineering culture slowly and unintentionally transmogrified from that adhering to a policy of "safety first" to "better, faster, cheaper." This change discouraged engineers from telling truth to power, including estimating the *actual* probability of shuttle-launch failure. Management needed the probability of launch failure to be less than 1 in 100,000 to allow launch. Any other answer was an annoyance and interfered with on-time and on-schedule launches. In an independent assessment, Richard Feynman found that when engineers were allowed to speak freely, they calculated the actual failure probability to be 1 in 100.[5] The engineering cultural failure killed many great and brave souls in two separate shuttle accidents.

I wrote *Engineering Trustworthy Systems* and this article to help enable and encourage engineers to take full charge of explicitly and intentionally managing system risk, from the ground up, in partnership with management and other key stakeholders.

Principles

It was no easy task to choose only 5% of the principles to discuss. When in doubt, I chose principles that may be less obvious to the reader, to pique curiosity and to attract more computer scientists and engineers to this important problem area. The ordering here is completely different than in the book so as to provide a logical flow of the presented subset.

Each primary principle includes a description of what the principle entails, a rationale for the creation of the principle, and a brief discussion of the implications on the cybersecurity discipline and its practice.

▸ **Cybersecurity's goal is to optimize mission effectiveness {03.01}.**

Description. Systems have a primary purpose or mission—to sell widgets, manage money, control chemical plants, manufacture parts, connect people, defend countries, fly airplanes, and so on. Systems generate mission value at a rate that is affected by the probability of failure from a multitude of causes, including cyberattack. The purpose of cybersecurity design is to reduce the probability of failure from cyberattack so as maximize mission effectiveness.

Rationale. Some cybersecurity engineers mistakenly believe that their goal is to maximize cybersecurity under a given budget constraint. This excessively narrow view misapprehends the nature of the engineering trade-offs with other aspects of system design and causes significant frustration among the cybersecurity designers, stakeholders in the mission system, and senior management (who must often adjudicate disputes between these teams). In reality, all teams are trying to optimize mission effectiveness. This realization places them in a collegial rather than an adversarial relationship.

Implications. Cybersecurity is always in a trade-off with mission functionality, performance, cost, ease-of-use and many other important factors. These trade-offs must be intentionally and explicitly managed. It is only in consideration of the bigger picture of optimizing mission that these trade-offs can be made in a reasoned manner.

▸ **Cybersecurity is about understanding and mitigating risk {02.01}.**

Description. Risk is the primary metric of cybersecurity. Therefore, understanding the nature and source of risk is key to applying and advancing the discipline. Risk measurement is foundational to improving cybersecurity {17.04}. Conceptually, cybersecurity risk is simply the probability of cyberattacks occurring multiplied by the potential damages that would result if they actually occurred. Estimating both of these quantities is challenging, but possible.

Rationale. Engineering disciplines require metrics to: "characterize the nature of what is and why it is that way, evaluate the quality of a system, predict system performance under a variety of environments and situations, and compare and improve systems continuously."[7] Without a metric, it is not possible to decide whether one system is better than another. Many fellow cybersecurity engineers complain that risk is difficult to measure and especially difficult to quantify, but proceeding without a metric is impossible. Thus, doing the hard work required to measure risk, with a reasonable uncertainty interval, is an essential part of the cybersecurity discipline. Sometimes, it seems that the cybersecurity community spends more energy complaining how difficult metrics are to create and measure accurately, than getting on with creating and measuring them.

Implications. With risk as the primary metric, risk-reduction becomes the primary value and benefit from any cybersecurity measure—technological or otherwise. Total cost of cybersecurity, on the other hand, is calculated in terms of the direct cost of procuring, deploying, and maintaining the cybersecurity mechanism as well as the indirect costs of mission impacts such as performance degradation, delay to market, capacity reductions, and usability. With risk-reduction as a benefit metric and an understanding of *total* costs, one can then reasonably compare alternate cybersecurity approaches in terms of risk-reduction return on investment. For example, it is often the case that there are no-brainer actions such as properly configuring existing security mechanisms (for example, firewalls and intrusion detection systems) that cost very little but significantly reduce the probability of successful cyberattack. Picking such low-hanging fruit should be the first step that any organization takes to improving their operational cybersecurity posture.

▸ **Theories of security come from theories of insecurity {02.03}.**

Description. One of the most important yet subtle aspects of an engineering discipline is understanding how to think about it—the underlying attitude that feeds insight. In the same way that failure motivates and informs dependability principles, cyberattack motivates and informs cybersecurity principles. Ideas on how to effectively defend a system, both during design and operation, must come from an understanding of how cyberattacks succeed.

Rationale. How does one prevent attacks if one does not know the mechanism by which attacks succeed? How does one detect attacks without knowing how attacks manifest? It is not possible. Thus, students of cybersecurity must be students of cyberattacks and adversarial behavior.

Implications. Cybersecurity engineers and practitioners should take courses and read books on ethical hacking. They should study cyberattack and particularly the post-attack analysis performed by experts and published or spoken about at conferences such as Black Hat and DEF CON. They should perform attacks within lab environments designed specifically to allow for safe experimentation. Lastly, when successful attacks do occur, cybersecurity analysts must closely study them for root causes and the implications to improved component design, improved operations, improved architecture, and improved policy. "Understanding failure is the key to success" {07.04}. For example, the five-whys analysis technique used by the National Transportation Safety Board (NTSB) to investigate aviation accidents[9] is useful to replicate and adapt to mining all the useful hard-earned defense information from the pain of a successful cyberattack.

▸ **Espionage, sabotage, and influence are goals underlying cyberattack {06.02}.**

Description. Understanding adversaries requires understanding their motivations and strategic goals. Adversaries have three basic categories of goals: espionage—stealing secrets to gain an unearned value or to destroy value by revealing stolen secrets; sabotage—hampering operations to slow progress, provide competitive advantage, or to destroy for ideological purposes; and, influence—affecting decisions and outcomes to favor an adversary's interests and goals, usually at the expense of those of the defender.

Rationale. Understanding the strategic goals of adversaries illuminates their value system. A value system suggests in which attack goals a potential adversary might invest most heavily in, and perhaps give insight into how they will pursue those goals. Different adversaries will place different weights on different goals within each of the three categories. Each will also be willing to spend different amounts to achieve their goals. Clearly, a nation-state intelligence organization, a transnational terrorist group, organized crime, a hacktivist and a misguided teenager trying to learn more about cyberattacks all have very different profiles with respect to these goals and their investment levels. These differences affect their respective behaviors with respect to different cybersecurity architectures.

Implications. In addition to informing the cybersecurity designer and operator (one who monitors status and controls the cybersecurity subsystem in real time), understanding attacker goals allows cybersecurity analysts to construct goal-oriented attack trees that are extraordinarily useful in guiding design and operation because they give insight into attack probability and attack sequencing. Attack sequencing, in turn, gives insight into getting ahead of attackers at interdiction points within the attack step sequencing {23.18}.

▸ **Assume your adversary knows your system well and is inside it {06.05}.**

Description. Secrecy is fleeting and thus should never be depended upon more than is absolutely necessary {03.05}. This is true of data but applies even more strongly with respect to the system itself {05.11}. It is unwise to make rash and unfounded assumptions that cannot be proven with regard to what a potential adversary may or may not know. It is much safer to assume they know at least as much as the designer does about the system. Beyond adversary knowledge of the system, a good designer makes the stronger assumption that an adversary has managed to co-opt at least part of the system sometime during its life cycle. It must be assumed that an adversary changed a component to have some degree of control over its function so as to operate as the adversary's inside agent.

Rationale. First, there are many opportunities for a system design and implementation to be exposed and subverted along its entire life cycle. Early development work is rarely protected very carefully. System components are often reused from previous projects or open source. Malicious changes can easily escape notice during system integration and testing because of the complexity of the software and hardware in modern systems. The maintenance and update phases are also vulnerable to both espionage and sabotage. The adversary also has an opportunity to stealthily study a system during operation by infiltrating and observing the system, learning how the system works in reality, not just how it was intended by the designer (which can be significantly different, especially after an appreciable time in operation). Second, the potential failure from making too weak of an assumption could be catastrophic to the system's mission, whereas making strong assumptions merely could make the system more expensive. Clearly, both probability (driven by opportunity) and prudence suggest making the more conservative assumptions.

It is much better to assume adversaries know at least as much as the designer does about the system.

Implications. The implications of assuming the adversary knows the system at least as well as the designers and operators are significant. This principle means that cybersecurity designers must spend a substantial amount of resources: Minimizing the probability of flaws in design and implementation through the design process itself, and performing extensive testing, including penetration and red-team testing focused specifically on looking at the system from an adversary perspective. The principle also implies a cybersecurity engineer must understand the residual risks in terms of any known weaknesses. The design must compensate for those weaknesses through architecture (for example, specifically focusing the intrusion detection system to monitor possible exploitation of those weaknesses), as opposed to hoping the adversary does not find them because they are "buried too deep" or, worse yet, because the defender believes that the attacker is "not that sophisticated." Underestimating the attacker is hubris. As the saying goes: pride comes before the fall {06.04}.

Assuming the attacker is (partially) inside the system requires the designer to create virtual bulkheads in the system and to detect and thwart attacks propagating from one part of the system (where the attacker may have a toehold) to the next. This is a wise approach because many sophisticated attacks, such as worms, often propagate within the system once they find their way in (for example, through a phishing attack on an unsuspecting user who clicked on an attacker's malicious link in an email message).

▸ **Without integrity, no other cybersecurity properties matter {03.06}.**

Description. Cybersecurity is sometimes characterized as having three pillars, using the mnemonic C-I-A: preserving *confidentiality* of data, ensuring the *integrity* of both the data and the system, and ensuring the *availability* of the system to provide the services for which it was designed. Sometimes, cybersecurity engineers become hyperfocused on one pillar to the exclusion of adequate attention to the others. This is particularly true of cybersecurity engineers who have their roots in U.S. Department of Defense (DoD) cybersecurity because confidentiality of classified data is a high-priority concern in the DoD. The reality is that all other system properties depend on system integrity, which therefore has primacy.

Rationale. System integrity is the single most important property because, without it, no other system properties are possible. No matter what properties a system may possess when deployed, they can be immediately subverted by the attacker altering the system to undo those properties and replace them with properties desirable to the attacker. This gives rise to the fundamental concept of the reference monitor {20.02}, which requires the security-critical subsystem be correct (perform the required security functions), non-bypassable (so that the attacker cannot circumvent the correct controls to access protected resources), and tamperproof (so the system cannot be altered without authorization).

Implications. This primacy-of-integrity principle means that cybersecurity engineers must focus attention on access control to the system as a first priority, including heavy monitoring of the system for any unauthorized changes. This priority extends to the earlier stages of system life cycle such as update distribution and maintenance.

The effectiveness of depth could be measured by how miserable it makes an attacker's life.

▸ **An attacker's priority target is the cybersecurity system {19.17}.**

Description. Closely following from the primacy-of-integrity principle {03.06} is the criticality of the cybersecurity subsystem. To attack the mission, it is necessary first to disable any security controls that effectively defend against the adversary's attack path—including the security controls that defend the security subsystem itself. Great care must be taken to protect and monitor the cybersecurity subsystem carefully {23.12}.

Rationale. The security subsystem protects the mission system. Therefore, attempted attacks on the cybersecurity subsystem are harbingers of attacks on the mission system itself {22.08}. The cybersecurity system is therefore a prime target of the adversary because it is the key to attacking the mission system. Protection of the cybersecurity system is thus paramount {21.03}. For example, the cybersecurity audit log integrity is important because attackers attempt to alter the log to hide evidence of their cyberattack activities.

Implications. The cybersecurity system must be carefully designed to itself be secure. The cybersecurity of the cybersecurity system cannot depend on any other less secure systems. Doing so creates an indirect avenue for attack. For example, if the identity and authentication process for access maintenance ports for updating the cybersecurity system use simple passwords over remotely accessible network ports, that becomes the weakest link of the entire system. In addition, cybersecurity engineers cannot simply use the cybersecurity mechanism that the cybersecurity system provides to protect the mission systems. In other words, the cybersecurity system cannot use itself to protect itself; that creates a circular dependency that will almost certainly create an exploitable flaw an attacker can use. Lastly, the cybersecurity mechanisms are usually hosted on operating systems and underlying hardware, which become the underbelly of the cybersecurity system. That underbelly must be secured using different cybersecurity mechanisms, and it is best if those mechanisms can be as simple as possible. Complexity is the enemy of cybersecurity because of the difficulty of arguing that complex systems are correct {19.09}.

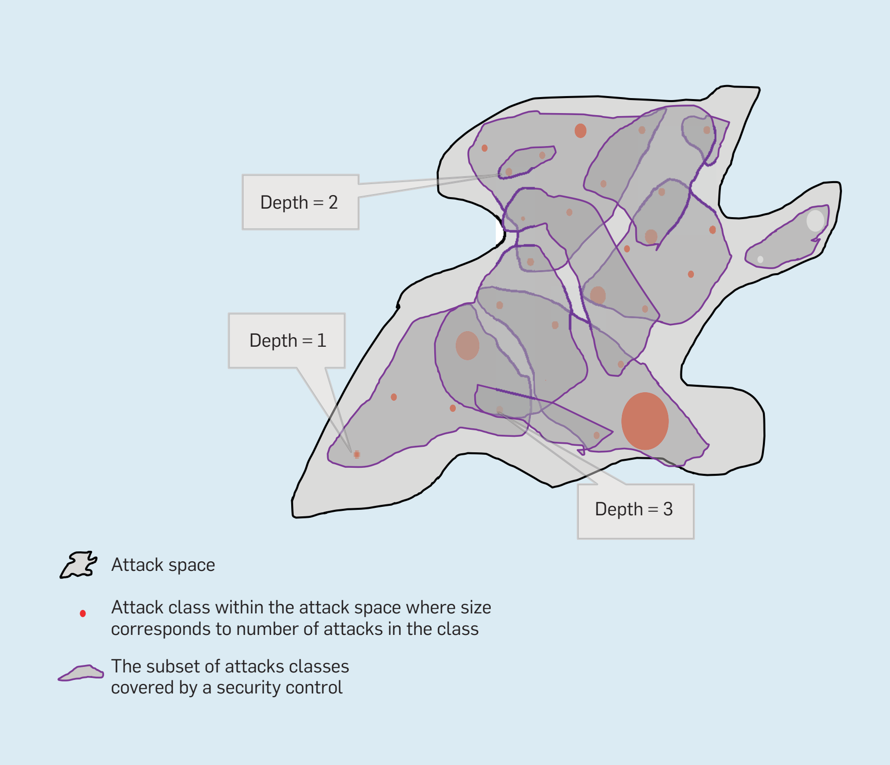
▸ **Depth without breadth is useless; breadth without depth, weak {08.02}.**

Description. Much ado has been made about the notion of the concept of *defense in depth*. The idea is often vaguely defined as layering cybersecurity approaches including people, diverse technology, and procedures to protect systems. Much more precision is needed for this concept to be truly useful to the cybersecurity design process. Layer how? With respect to what? The unspoken answer is the cyberattack space that covers the gamut of all possible attack classes as shown in the accompanying figure.

Rationale. One must achieve depth with respect to specified attack classes. Mechanisms that are useful against some attack classes are entirely useless against others. This focusing idea fosters an equally important companion principle: *defense in breadth*. If a cybersecurity designer creates excellent depth to the point of making a particular class of attack prohibitive to an adversary, the adversary may simply move to an alternative attack. Thus, *one must cover the breadth of the attack space, in depth*. Ideally, the depth will be such that all avenues of attack, for all attack classes, will be equally difficult, and above the cost and risk thresholds of the attackers.

Implications. This depth-and-breadth principle implies that the cybersecurity engineer must have a firm understanding of the entire spectrum of cyberattacks, not just a few attacks. More broadly, the principle suggests the cybersecurity community must develop better cyberattack taxonomies that capture the entire attack space, including hardware attacks, device controller attacks, operating system attacks, and cyberattacks used to affect the beliefs of people. Further, the principle also means that cybersecurity measures must be properly characterized in terms of their effectiveness against the various portions of the cyberattack space. Those who create or advocate for various measures or solutions will be responsible for creating specific claims about their cyberattack-space coverage, and analysts will be responsible for designing tests to thoroughly evaluate the validity of those claims. Lastly, cybersecurity architects will need to develop techniques for weaving together cybersecurity in ways that create true depth, measured by how the layers alter the probability of success an adversary will have for the targeted attack class. Said a different way, *the effectiveness of depth could be measured by how miserable it makes an attacker's life.*

Defense depth and breadth in a cyberattack.


Depth = 2
Depth = 1
Depth = 3
Attack space
Attack class within the attack space where size corresponds to number of attacks in the class
The subset of attacks classes covered by a security control

▸ **Failing to plan for failure guarantees catastrophic failure {20.06}.**

Description. System failures are inevitable {19.01, 19.05}. Pretending otherwise is almost always catastrophic. This principle applies to both the mission system and cybersecurity subsystem that protects the mission system. Cybersecurity engineers must understand that their systems, like all systems, are subject to failure. It is incumbent on those engineers to understand how their systems can possibly fail, including the failure of the underlying hardware and other systems on which they depend (forexample, the microprocessors, the internal system bus, the network, memory, and external storage systems). A student of cybersecurity is a student of failure {07.01} and thus a student of dependability as a closely related discipline. Security requires reliability; reliability requires security {05.09}.

Rationale. Too many cybersecurity engineers forget that cybersecurity mechanisms are not endowed with magical powers of nonfailure. Requirements can be ambiguous and poorly interpreted, designs can be flawed, and implementation errors are no less likely in security code than in other code. Indeed, security code often has to handle complex timing issues and sometimes needs to be involved in hardware control. This involves significantly more complexity than normal systems and thus requires even more attention to failure avoidance, detection, and recovery {05.10}. Yet the average cybersecurity engineer today seems inadequately schooled in this important related discipline.

Implications. Cybersecurity engineering requires design using dependability engineering principles. This means that cybersecurity engineers must understand the nature and cause of faults, how the activation of faults lead to errors, which can propagate and cause system failures.[1] They must understand this not only with respect to the cybersecurity system they design, but all the systems on which the system depends and which depend on it, including the mission system itself.

▸ **Strategy and tactics knowledge**

comes from attack encounters {01.09}.

Description. As important as good cybersecurity design is, good cybersecurity operations is at least as important. Each cybersecurity mechanism is usually highly configurable with hundreds, thousands, and even millions of possible settings (for example, the rule set of firewalls denying or permitting each combination, port, protocol, source address range, and destination address range). What are the optimal settings of all of these various mechanisms? The answer depends on variations in the mission and variations in the system environment, including attack attempts that may be ongoing. The settings are part of a trade-off space for addressing the entire spectrum of attacks. The reality is there is no static optimal setting for all cyberattack scenarios under all possible conditions {22.07}. Furthermore, dynamically setting the controls leads to a complex control-feedback problem {23.11}. Where does the knowledge come from regarding how to set the security control parameters according to the particulars of the current situation? It is extracted from the information that comes from analyzing cyberattack encounters, both real and simulated, both those that happen to one's own organization and those that happen to one's neighbors.

Rationale. There is certainly good theory, such as game-theory based approaches,[2] which one can develop about how to control the system effectively (for example, using standard control theory). On the other hand, practical experience plays an important role in learning how to effectively defend a system. This knowledge is called strategy (establishing high-level goals in a variety of different situations) and tactics (establishing effective near-term responses to attack steps the adversary takes).

Implications. Strategy and tactics knowledge must be actively sought, collected with intention (through analyzing real encounters, performing controlled experiments, and performing simulations {23.04}), curated, and effectively employed in the operations of a system. Cybersecurity systems must be designed to store, communicate, and use this knowledge effectively in the course of real operations. Plans based on this knowledge are sometimes called playbooks. They must be developed in advance of attacks {23.05} and must be broad enough {23.07} to handle a large variety of attack situations that are likely to occur in real-world operations. The process of thinking through responses to various cyberattack scenarios, in itself, is invaluable in the planning process {23.10}. Certain responses that may be contemplated during this process may need infrastructure (such as, actuators) to execute the action accurately and quickly enough {23.15} to be effective. This insight will likely lead to design requirements for implementing such actuators as the system is improved.

The Future

Systematically extracting, presenting, and building the principles underlying trustworthy systems design is not the work of one cybersecurity engineer—not by a long shot. The task is difficult, daunting, complex, and never-ending. I mean here to present a beginning, not the last word on the matter. My goal is to encourage the formation of a community of cybersecurity and systems engineers strongly interested in maturing and advancing their discipline so that others may stand on their shoulders. This community is served by like-minded professionals sharing their thoughts, experiences, and results in papers, conferences, and over a beverage during informal gatherings. My book and this article are a call to action for this community to organize and work together toward the lofty goal of building the important underpinnings from a systems-engineering perspective.

Lastly, I will point out that cyberattack measures and cybersecurity countermeasures are in an eternal co-evolution and co-escalation {14.01}. Improvements to one discipline will inevitably create an evolutionary pressure on the other. This has at least two important implications. First, the need to build cybersecurity knowledge to build and operate trustworthy systems will need continuous and eternal vigilant attention. Second, communities on both sides need to be careful about where the co-evolution leads. Faster and faster cyberattacks will lead cybersecurity defenders to autonomic action and planning that may eventually be driven by artificial intelligence. Stronger and stronger cybersecurity measures that dynamically adapt to cyberattacks will similarly lead adversaries to more intelligent and autonomic adaptations in their cyberattacks. The road inevitably leads to machine-controlled autonomic action-counteraction and machine-driven adaptation and evolution of mechanisms. This may have surprising and potentially disastrous results to the system called humanity {25.02, 25.04}.

Acknowledgments

First and foremost, I acknowledge all of the formative conversations with my technical mentor, Brian Snow. He is a founding cybersecurity intellectual who has generously, gently, and wisely guided many minds throughout his illustrious career. Second, I thank the dozens of brilliant cybersecurity engineers and scientists with whom I have had the opportunity to work over the last three decades. Each has shone a light of insight from a different direction that helped me see the bigger picture of underlying principles. C

References

1. Avizienis, A., Laprie, J.-C., and Randell, B. Fundamental concepts of dependability. In *Proceedings of the 3rd IEEE Information Survivability Workshop* (Boston, MA, Oct. 24–26). IEEE, 2000, 7–12.
2. Hamilton, S.N., Miller, W.L., Ott, A., and Saydjari, O.S. The role of game theory in information warfare. In *Proceedings of the 4th Information Survivability Workshop.* 2001.
3. Hammond, S.A. and Mayfield, A.B. *The Thin Book of Naming Elephants: How to Surface Undiscussables for Greater Organizational Success.* McGraw-Hill, New York, 2004, 290–292.
4. Morgan, S. *Top 5 Cybersecurity Facts, Figures and Statistics for 2018.* CSO Online; https://bit.ly/2KG6jJV.
5. NASA. *Report of the Presidential Commission on the Space Shuttle Challenger Accident.* June 6, 1986; https://history.nasa.gov/rogersrep/genindex.htm
6. Rand Corporation. *Foundations of Effective Influence Operations: A Framework for Enhancing Army Capabilities.* Rand Corp. 2009; https://www.rand.org/content/dam/rand/pubs/monographs/2009/RAND_MG654.pdf
7. Saydjari, O.S. *Why Measure? Engineering Trustworthy Systems.* McGraw-Hill, New York, 2018, 290–292.
8. Saydjari, O.S. *Engineering Trustworthy Systems: Get Cybersecurity Design Right the First Time.* McGraw-Hill Education, 2018.
9. Wiegmann, D. and Shappell, S.A. *A Human Error Approach to Aviation Accident Analysis: The Human Factors Analysis and Classification System.* Ashgate Publishing, 2003.
10. Zarate, J.C. The Cyber Attacks on Democracy. *The Catalyst 8*, (Fall 2017); https://bit.ly/2IXttZr

O. Sami Saydjari (ssaydjari@gmail.com) is Founder and President of the Cyber Defense Agency, Inc., Clarksville, MD, USA.

DOI:10.1145/3282486

To trust the behavior of complex AI algorithms, especially in mission-critical settings, they must be made intelligible.

BY DANIEL S. WELD AND GAGAN BANSAL

The Challenge of Crafting Intelligible Intelligence

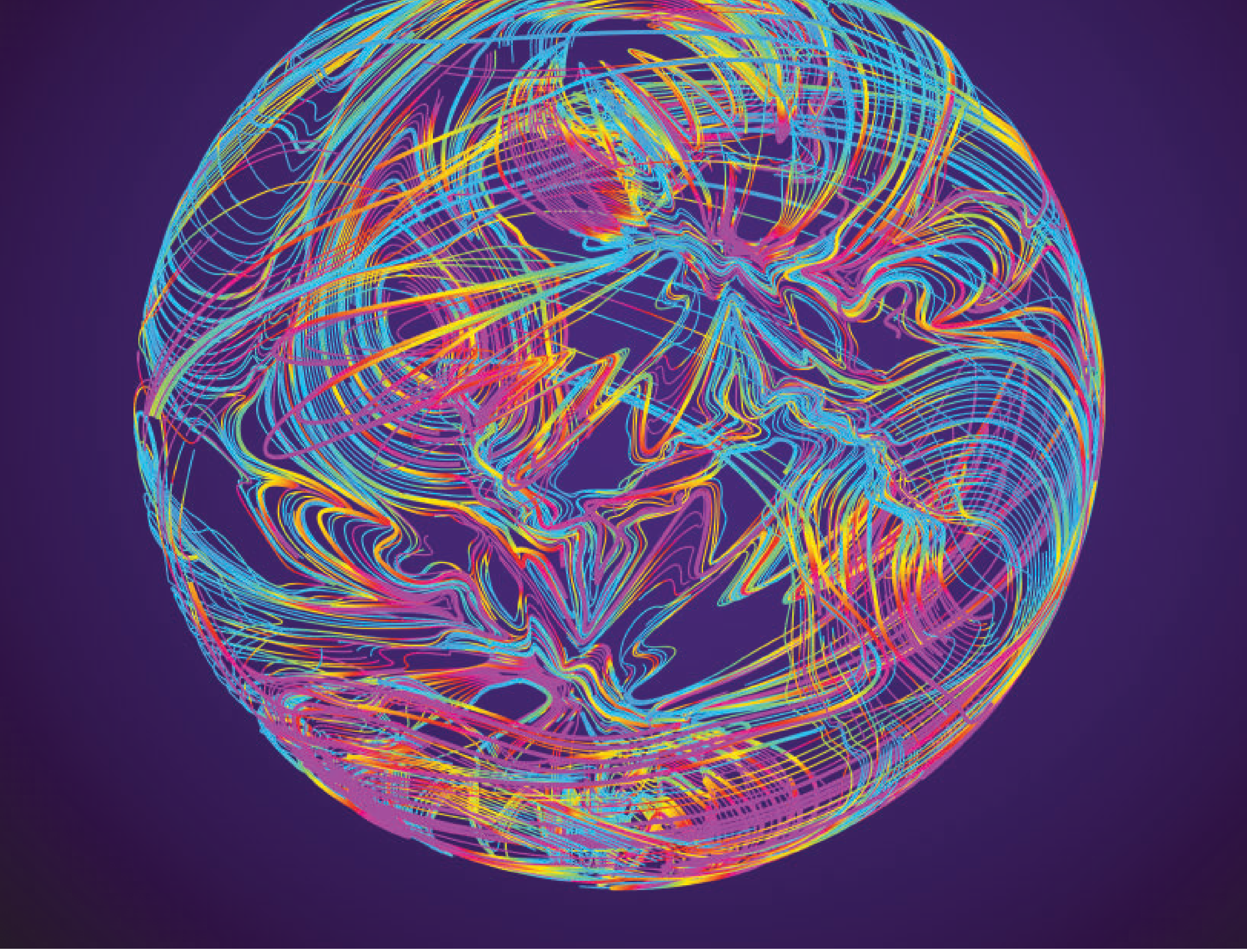
ARTIFICIAL INTELLIGENCE (AI) systems have reached or exceeded human performance for many circumscribed tasks. As a result, they are increasingly deployed in mission-critical roles, such as credit scoring, predicting if a bail candidate will commit another crime, selecting the news we read on social networks, and self-driving cars. Unlike other mission-critical software, extraordinarily complex AI systems are difficult to test: AI decisions are context specific and often based on thousands or millions of factors. Typically, AI behaviors are generated by searching vast action spaces or learned by the opaque optimization of mammoth neural networks operating over prodigious amounts of training data. Almost by definition, no clear-cut method can accomplish these AI tasks.

Unfortunately, much AI-produced behavior is alien, that is, it can fail in unexpected ways. This lesson is most clearly seen in the performance of the latest deep neural network image analysis systems. While their accuracy at object-recognition on naturally occurring pictures is extraordinary, imperceptible changes to input images can lead to erratic predictions, as shown in Figure 1. Why are these recognition systems so brittle, making different predictions for apparently identical images? Unintelligible behavior is not limited to machine learning; many AI programs, such as automated planning algorithms, perform search-based look ahead and inference whose complexity exceeds human abilities to verify. While some search and planning algorithms are provably complete and optimal, intelligibility is still important, because the underlying primitives (for example, search operators or action descriptions) are usually approximations.[29] One can't trust a proof that is based on (possibly) incorrect premises.

Despite intelligibility's apparent value, it remains remarkably difficult to specify what makes a system "intelligible." (We discuss desiderata for intelligible behavior later in this article.) In brief, we seek AI systems where it is clear what factors caused the system's action,[24] allowing the users to predict how changes to the situation would have led to alternative behaviors, and permits effective control of

» key insights

- There are important technical and social reasons to prefer inherently intelligible AI models (such as linear models or GA^{2}Ms) over deep neural models; furthermore, intelligible models often have comparable accuracy.
- When an AI system is based on an inscrutable model, it may explain its decisions by mapping those decisions onto a simpler, explanatory model using techniques such as local approximation and vocabulary transformation.
- Results from psychology show that explanation is a process, best thought of as a conversation between explainer and listener. We advocate increased work on interactive explanation systems that can respond to a wide range of follow-up questions.



the AI by enabling interaction. As we will illustrate, there is a central tension between a concise explanation and an accurate one.

As shown in Figure 2, our survey focuses on two high-level approaches to building intelligible AI software: ensuring the underlying reasoning or learned model is inherently interpretable, for example, by learning a linear model over a small number of well-understood features, and if it is necessary to use an inscrutable model, such as complex neural networks or deep-look ahead search, then mapping this complex system to a simpler, explanatory model for understanding and control.[28] Using an interpretable model provides the benefit of transparency and veracity; in theory, a user can see exactly what the model is doing. Unfortunately, interpretable methods may not perform as well as more complex ones, such as deep neural networks. Conversely, the approach of mapping to an explanatory model can apply to whichever AI technique is currently delivering the best performance, but its explanation inherently differs from the way the AI system actually operates. This yields a central conundrum: How can a user trust that such an explanation reflects the essence of the underlying decision and does not conceal important details? We posit the answer is to make the explanation system interactive so users can drill down until they are satisfied with their understanding.

The key challenge for designing intelligible AI is communicating a complex computational process to a human. This requires interdisciplinary skills, including HCI as well as AI and machine learning expertise. Furthermore, since the nature of explanation has long been studied by philosophy and psychology, these fields should also be consulted.

This article highlights key approaches and challenges for building intelligible intelligence, characterizes intelligibility, and explains why it is important even in systems with measurably high performance. We describe the benefits and limitations of GA^2M—a powerful class of interpretable ML models. Then, we characterize methods for handling inscrutable models, discussing different strategies for mapping to a simpler, intelligible model appropriate for explanation and control. We sketch a vision for building interactive explanation systems, where the mapping changes in response to the user's needs. Lastly, we argue that intelligibility is important for search-based AI systems as well as for those based on machine learning and that similar solutions may be applied.

IMAGE FROM SHUTTERSTOCK.COM

Why Intelligibility Matters

While it has been argued that explanations are much less important than sheer performance in AI systems, there are many reasons why intelligibility is important. We start by discussing technical reasons, but social factors are important as well.

AI may have the wrong objective. In some situations, even 100% perfect performance may be insufficient, for example, if the performance metric is flawed or incomplete due to the difficulty of specifying it explicitly. Pundits have warned that an automated factory charged with maximizing paperclip production, could subgoal on killing humans, who are using resources that could otherwise be used in its task. While this example may be fanciful, it illustrates that it is remarkably difficult to balance multiple attributes of a utility function. For example, as Lipton observed,[25] "An algorithm for making hiring decisions should simultaneously optimize for productivity, ethics and legality." However, how does one express this trade-off? Other examples include balancing training error while uncovering causality in medicine and balancing accuracy and fairness in recidivism prediction.[12] For the latter, a simplified objective function such as accuracy combined with historically biased training data may cause uneven performance for different groups (for example, people of color). Intelligibility empowers users to determine if an AI is right for the right reasons.

Figure 1. Adding an imperceptibly small vector to an image changes the GoogLeNet[39] image recognizer's classification of the image from "panda" to "gibbon." Source: Goodfellow et al.[9]

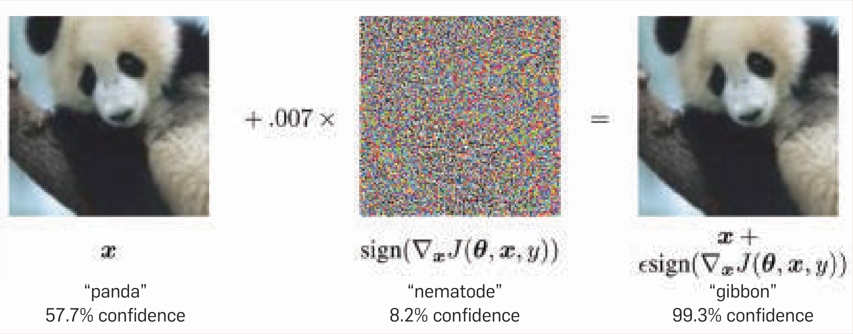

+ .007 ×
=
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\,\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3% confidence

Figure 2. Approaches for crafting intelligible AI.

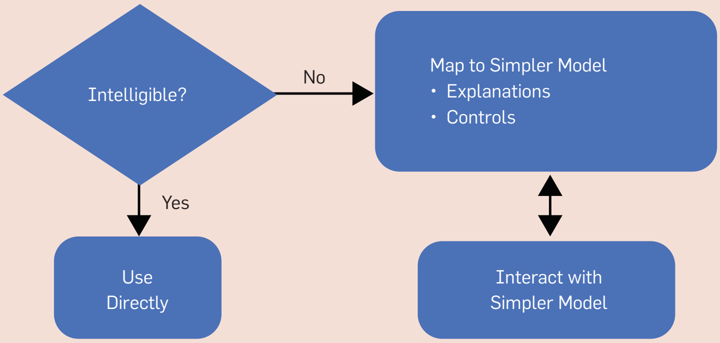
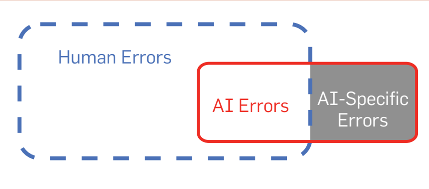


Figure 3. The dashed blue shape indicates the space of possible mistakes humans can make.

The red shape denotes the AI's mistakes; its smaller size indicates a net reduction in the number of errors. The gray region denotes AI-specific mistakes a human would never make. Despite reducing the total number of errors, a deployed model may create new areas of liability (gray), necessitating explanations.



AI may be using inadequate features. Features are often correlated, and when one feature is included in a model, machine learning algorithms extract as much signal as possible from it, indirectly modeling other features that were not included. This can lead to problematic models, as illustrated by Figure 4b (and described later), where the ML determined that a patient's prior history of asthma (a lung disease) was negatively correlated with death by pneumonia, presumably due to correlation with (unmodeled) variables, such as these patients receiving timely and aggressive therapy for lung problems. An intelligible model helps humans to spot these issues and correct them, for example, by adding additional features.[4]

Distributional drift. A deployed model may perform poorly in the wild, that is, when a difference exists between the distribution which was used during training and that encountered during deployment. Furthermore, the deployment distribution may change over time, perhaps due to feedback from the act of deployment. This is common in adversarial domains, such as spam detection, online ad pricing, and search engine optimization. Intelligibility helps users determine when models are failing to generalize.

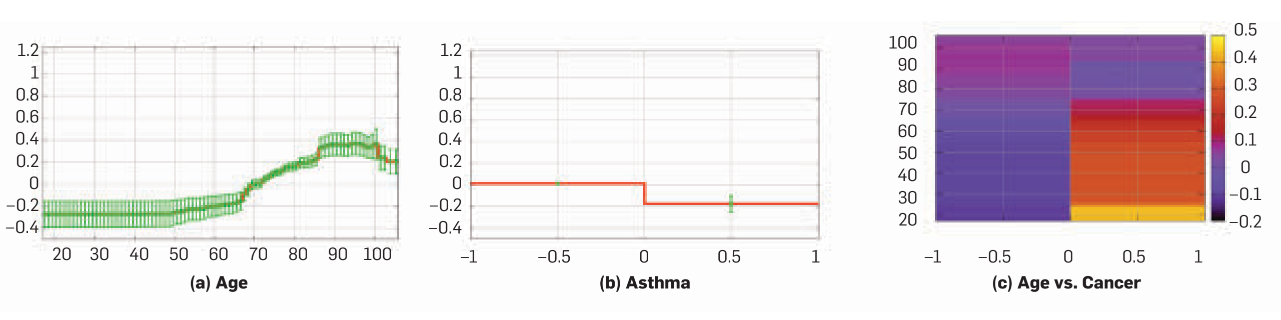
Facilitating user control. Many AI systems induce user preferences from their actions. For example, adaptive news feeds predict which stories are likely most interesting to a user. As robots become more common and enter the home, preference learning will become ever more common. If users understand why the AI performed an undesired action, they can better issue instructions that will lead to improved future behavior.

User acceptance. Even if they do not seek to change system behavior, users have been shown to be happier with and more likely to accept algorithmic decisions if they are accompanied by an explanation.[18] After being told they should have their kidney removed, it's natural for a patient to ask the doctor why—even if they don't fully understand the answer.

Improving human insight. While improved AI allows automation of tasks previously performed by humans, this is not their only use. In ad-

Figure 4. A part of Figure 1 from Caruana et al.[4] showing three (of 56 total) components for a GA^{2}M model, which was trained to predict a patient's risk of dying from pneumonia.

The two line graphs depict the contribution of individual features to risk: patient's age, and Boolean variable asthma. The *y*-axis denotes its contribution (log odds) to predicted risk. The heat map visualizes the contribution due to pairwise interactions between age and cancer rate.



dition, scientists use machine learning to get insight from big data. Medicine offers several examples.[4] Similarly, the behavior of AlphaGo[35] has revolutionized human understanding of the game. Intelligible models greatly facilitate these processes.

Legal imperatives. The European Union's GDPR legislation decrees citizens' right to an explanation, and other nations may follow. Furthermore, assessing legal liability is a growing area of concern; a deployed model (for example, self-driving cars) may introduce new areas of liability by causing accidents unexpected from a human operator, shown as "AI-specific error" in Figure 3. Auditing such situations to assess liability requires understanding the model's decisions.

Defining Intelligibility

So far we have treated intelligibility informally. Indeed, few computing researchers have tried to formally define what makes an AI system interpretable, transparent, or intelligible,[6] but one suggested criterion is human simulatability:[25] Can a human user easily predict the model's output for a given input? By this definition, sparse linear models are more interpretable than dense or non-linear ones.

Philosophers, such as Hempel and Salmon, have long debated the nature of explanation. Lewis[23] summarizes: "To explain an event is to provide some information about its causal history." But many causal explanations may exist. The fact that event C causes E is best understood relative to an imagined counterfactual scenario, where absent C, E would not have occurred; furthermore, C should be minimal, an intuition known to early scientists, such as William of Occam, and formalized by Halpern and Pearl.[11]

Following this logic, we suggest a better criterion than simulatability is the ability to answer counterfactuals, aka "what-if" questions. Specifically, we say that a model is intelligible to the degree that a human user can predict how a change to a feature, for example, a small increase to its value, will change the model's output and if they can reliably modify that response curve. Note that if one can simulate the model, predicting its output, then one can predict the effect of a change, but not vice versa.

Linear models are especially interpretable under this definition because they allow the answering of counterfactuals. For example, consider a naive Bayes unigram model for sentiment analysis, whose objective is to predict the emotional polarity (positive or negative) of a textual passage. Even if the model were large, combining evidence from the presence of thousands of words, one could see the effect of a given word by looking at the sign and magnitude of the corresponding weight. This answers the question, "What if the word had been omitted?" Similarly, by comparing the weights associated with two words, one could predict the effect on the model of substituting one for the other.

Ranking intelligible models. Since one may have a choice of intelligible models, it is useful to consider what makes one preferable to another. Social science research suggests an explanation is best considered a social process, a conversation between explainer and explainee.[15,30] As a result, Grice's rules for cooperative communication[10] may hold for intelligible explanations. Grice's maxim of quality says be truthful, only relating things that are supported by evidence. The maxim of quantity says to give as much information as is needed, and no more. The maxim of relation: only say things that are relevant to the discussion. The maxim of manner says to avoid ambiguity, being as clear as possible.

Miller summarizes decades of work by psychological research, noting that explanations are contrastive, that is, of the form "Why P rather than Q?" The event in question, P, is termed the fact and Q is called the foil.[30] Often the foil is not explicitly stated even though it is crucially important to the explanation process. For example, consider the question, "Why did you predict the image depicts an indigo bunting?" An explanation that points to the color blue implicitly assumes the foil is another bird, such as a chickadee. But perhaps the questioner wonders why the recognizer did not predict a pair of denim pants; in this case a more precise explanation might highlight the presence of wings and a beak. Clearly, an explanation targeted to the wrong foil will be unsatisfying, but the nature and sophistication of a foil can depend on the end user's expertise; hence, the ideal explanation will differ for different people.[6] For example, to verify that an ML system is fair, an ethicist might generate more complex foils than a data scientist. Most ML explanation systems have restricted their attention to elucidating the behavior of a binary classifier, that is, where there is only one possible foil choice. However, as we seek to explain multiclass systems, addressing this issue becomes essential.

Many systems are simply too complex to understand without approximation. Here, the key challenge is deciding which details to omit. After many years of study, psychologists determined that several criteria can be prioritized for inclusion in an explanation: necessary causes (vs. sufficient ones); intentional actions (vs. those taken without deliberation); proximal causes (vs. distant ones); details that distinguish between fact and foil; and abnormal features.[30]

According to Lombrozo, humans prefer explanations that are simpler (that is, contain fewer clauses), more general, and coherent (that is, consistent with what the human's prior beliefs).[26] In particular, she observed the surprising result that humans preferred simple (one clause) explanations to conjunctive ones, even when the probability of the latter was higher than the former.[26] These results raise interesting questions about the purpose of explanations in an AI system. Is an explanation's primary purpose to convince a human to accept the computer's conclusions (perhaps by presenting a simple, plausible, but unlikely explanation) or is it to educate the human about the most likely true situation? Tversky, Kahneman, and other psychologists have documented many cognitive biases that lead humans to incorrect conclusions; for example, people reason incorrectly about the probability of conjunctions, with a concrete and vivid scenario deemed more likely than an abstract one that strictly subsumes it.[16] Should an explanation system exploit human limitations or seek to protect us from them?

Other studies raise an additional complication about how to communicate a system's uncertain predictions to human users. Koehler found that simply presenting an explanation for a proposition makes people think that it is more likely to be true.[18] Furthermore, explaining a fact in the same way as previous facts have been explained amplifies this effect.[36]

Inherently Intelligible Models

Several AI systems are inherently intelligible, and we previously observed that linear models support counterfactual reasoning. Unfortunately, linear models have limited utility because they often result in poor accuracy. More expressive choices may include simple decision trees and compact decision lists. To concretely illustrate the benefits of intelligibility, we focus on Generalized additive models (GAMs), which are a powerful class of ML models that relate a set of features to the target using a linear combination of (potentially nonlinear) single-feature models called shape functions.[27] For example, if y represents the target and $\{x_1, ...x_n\}$ represents the features, then a GAM model takes the form $y = \beta_0 + \Sigma_j f_j\,(x_j)$, where the f_is denote shape functions and the target y is computed by summing single-feature terms. Popular shape functions include non-linear functions such as splines and decision trees. With linear shape functions GAMs reduce to a linear models. GA^{2}M models extend GAM models by including terms for pairwise interactions between features:

$$y = \beta_0 + \sum_j f_j(x_j) + \underbrace{\sum_{i\neq j} f_{ij}(x_i, x_j)}_{\text{pairwise terms}} \tag{1}$$

Caruana et al. observed that for domains containing a moderate number of semantic features, GA^{2}M models achieve performance that is competitive with inscrutable models, such as random forests and neural networks, while remaining intelligible.[4] Lou et al. observed that among methods available for learning GA^{2}M models, the version with bagged shallow regression tree shape functions learned via gradient boosting achieves the highest accuracy.[27]

Both GAM and GA^{2}M are considered interpretable because the model's learned behavior can be easily understood by examining or visualizing the contribution of terms (individual or pairs of features) to the final prediction. For example, Figure 4 depicts a GA^{2}M model trained to predict a patient's risk of dying due to pneumonia, showing the contribution (log odds) to total risk for a subset of terms. A positive contribution increases risk, whereas a negative contribution decreases risk. For example, Figure 4a shows how the patient's age affects predicted risk. While the risk is low and steady for young patients (for example, age < 20), it increases rapidly for older patients (age > 67). Interestingly, the model shows a sudden increase at age 86; perhaps a result of less aggressive care by doctors for patients "whose time has come." Even more surprising is the sudden drop for patients over 100. This might be another social effect; once a patient reaches the magic "100," he or she gets more aggressive care. One benefit of an interpretable model is its ability to highlight these issues, spurring deeper analysis.

Figure 4b illustrates another surprising aspect of the learned model; apparently, a history of asthma, a respiratory disease, *decreases* the patients risk of dying from pneumonia! This finding is counterintuitive to any physician, who recognizes that asthma, in fact, should in theory increase such risk. When Caruana et al. checked the data, they concluded the lower risk was likely due to correlated variables—asthma patients typically receive timely and aggressive therapy for lung issues. Therefore, although the model was highly accurate on the test set, it would likely fail, dramatically underestimating the risk to a patient with asthma who had not been previously treated for the disease.

Facilitating human control of GA^{2}M models. A domain expert can fix such erroneous patterns learned by the model by setting the weight of the asthma term to zero. In fact, GA^{2}Ms let users provide much more comprehensive feedback to the model by using a GUI to redraw a line graph for model terms.[4] An alternative remedy might be to introduce a new feature to the model, representing whether the patient had been recently seen by a pulmonologist. After adding this feature, which is highly correlated with asthma, and retraining, the newly learned model would likely reflect that asthma (by itself) increases the risk of dying from pneumonia.

There are two more takeaways from this anecdote. First, the absence of an important feature in the data representation can cause any AI system to learn unintuitive behavior for another, correlated feature. Second, if the learner is intelligible, then this unintuitive behavior is immediately apparent, allowing appropriate skepticism (despite high test accuracy) and easier debugging.

Recall that GA^{2}Ms are more expressive than simple GAMs because they include pairwise terms. Figure 4c depicts such a term for the features age and cancer. This explanation indicates that among the patients who have cancer, the younger ones are at higher risk. This may be because the younger patients who develop cancer are probably critically ill. Again, since doctors can readily inspect these terms, they know if the learner develops unexpected conclusions.

Limitations. As described, GA^{2}M models are restricted to binary classification, and so explanations are clearly contrastive—there is only one choice of foil. One could extend GA^{2}M to handle multiple classes by training *n* one-vs-rest classifiers or building a hierarchy of classifiers. However, while these approaches would yield a working multi-class classifier, we don't know if they preserve model intelligibility, nor whether a user could effectively adjust such a model by editing the shape functions.

Furthermore, recall that GA^{2}Ms decompose their prediction into effects of individual terms, which can be visualized. However, if users are confused about what terms mean, they will not understand the model or be able to ask meaningful "what-if" questions. Moreover, if there are too many features, the model's complexity may be overwhelming. Lipton notes that the effort required to simulate some models (such as decision trees) may grow logarithmically with the number of parameters,[25] but for GA^{2}M the number of visualizations to inspect could increase quadratically. Several methods might help users manage this complexity; for example, the terms could be ordered by importance; however, it's not clear how to estimate importance. Possible methods include using an ablation analysis to compute influence of terms on model performance or computing the maximum contribution of terms as seen in the training samples. Alternatively, a domain expert could group terms semantically to facilitate perusal.

However, when the number of features grows into the millions—which occur when dealing with classifiers over text, audio, image, and video data—existing intelligible models do not perform nearly as well as inscrutable methods, like deep neural networks. Since these models combine millions of features in complex, nonlinear ways, they are beyond human capacity to simulate.

The key challenge for designing intelligible AI is communicating a complex computational process to a human.

Understanding Inscrutable Models

There are two ways that an AI model may be inscrutable. It may be provided as a blackbox API, such as Microsoft Cognitive Services, which uses machine learning to provide image-recognition capabilities but does not allow inspection of the underlying model. Alternatively, the model may be under the user's control yet extremely complex, such as a deep, neural network, where a user has access to myriad learned parameters but cannot reasonably interpret them. How can one best explain such models to the user?

The comprehensibility/fidelity tradeoff. A good explanation of an event is both *easy to understand* and *faithful*, conveying the true cause of the event. Unfortunately, these two criteria almost always conflict. Consider the predictions of a deep neural network with millions of nodes: a complete and accurate trace of the network's prediction would be far too complex to understand, but any simplification sacrifices faithfulness.

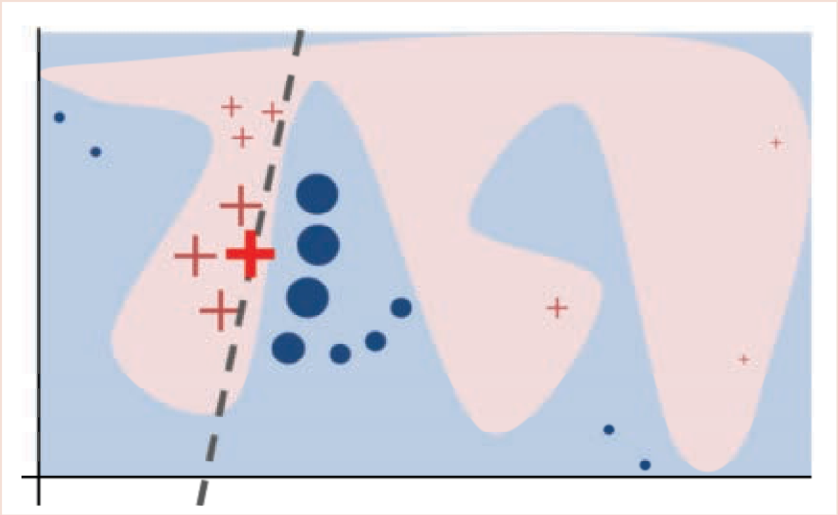
Finding a satisfying explanation, therefore, requires balancing the competing goals of comprehensibility and fidelity. Lakkaraju et al.[22] suggest formulating an explicit optimization of this form and propose an approximation algorithm for generating global explanations in the form of compact sets of if-then rules. Ribeiro et al. describe a similar optimization algorithm that balances faithfulness and coverage in its search for summary rules.[34]

Indeed, all methods for rendering an inscrutable model intelligible require mapping the complex model to a simpler one.[28] Several high-level approaches to mapping have been proposed.

Local explanations. One way to simplify the explanation of a learned model is to make it relative to a single input query. Such explanations, which are termed local[33] or *instance-based*,[22]

Figure 5. The intuition guiding LIME's method for constructing an approximate local explanation. Source: Ribeiro et al.[33]

"The black-box model's complex decision function, *f*, (unknown to LIME) is represented by the blue/pink background, which cannot be approximated well by a linear model. The bold red cross is the instance being explained. LIME samples instances, gets predictions using *f*, and weighs them by the proximity to the instance being explained (represented here by size). The dashed line is the learned explanation that is locally (but not globally) faithful."



are akin to a doctor explaining specific reasons for a patient's diagnosis rather than communicating all of her medical knowledge. Contrast this approach with the global understanding of the model that one gets with a GA^{2}M model. Mathematically, one can see a local explanation as currying—several variables in the model are fixed to specific values, allowing simplification.

Generating a local explanation is a common practice in AI systems. For example, early rule-based expert systems included explanation systems that augmented a trace of the system's reasoning—for a particular case—with background knowledge.[38] Recommender systems, one of the first deployed uses of machine learning, also induced demand for explanations of their specific recommendations; the most satisfying answers combined justifications based on the user's previous choices, ratings of similar users, and features of the items being recommended.[32]

Locally approximate explanations. In many cases, however, even a local explanation can be too complex to understand without approximation. Here, the key challenge is deciding which details to omit when creating the simpler explanatory model. Human preferences, discovered by psychologists and summarized previously, should guide algorithms that construct these simplifications.

Ribeiro et al.'s LIME system[33] is a good example of a system for generating a locally approximate explanatory model of an arbitrary learned model, but it sidesteps part of the question of which details to omit. Instead, LIME requires the developer to provide two additional inputs: A set of semantically meaningful features X' that can be computed from the original features, and an interpretable learning algorithm, such as a linear classifier (or a GA^{2}M), which it uses to generate an explanation in terms of the X'.

The insight behind LIME is shown in Figure 5. Given an instance to explain, shown as the bolded red cross, LIME randomly generates a set of similar instances and uses the blackbox classifier, *f*, to predict their values (shown as the red crosses and blue circles). These predictions are weighted by their similarity to the input instance (akin to *locally weighted regression*) and used to train a new, simpler intelligible classifier, shown on the figure as the linear decision boundary, using X', the smaller set of semantic features. The user receives the intelligible classifier as an explanation. While this explanation model[28] is likely a poor global representation of *f*, it is hopefully an accurate local approximation of the boundary in the vicinity of the instance being explained.

Ribeiro et al. tested LIME on several domains. For example, they explained the predictions of a convolutional neural network image classifier by converting the pixel-level features into a smaller set of "super-pixels;" to do so, they ran an off-the-shelf segmentation algorithm that identified regions in the input image and varied the color of some these regions when generating "similar" images. While LIME provides no formal guarantees about its explanations, studies showed that LIME's explanations helped users evaluate which of several classifiers best generalizes.

Choice of explanatory vocabulary. Ribeiro et al.'s use of presegmented image regions to explain image classification decisions illustrates the larger problem of determining an explanatory vocabulary. Clearly, it would not make sense to try to identify the exact pixel that led to the decision: pixels are too low level a representation and are not semantically meaningful to users. In fact, deep neural network's power comes from the very fact that their hidden layers are trained to recognize latent features in a manner that seems to perform much better than previous efforts to define such features independently. Deep networks are inscrutable exactly because we do not know what those hidden features denote.

To explain the behavior of such models, however, we must find some high-level abstraction over the input pixels that communicate the model's essence. Ribeiro et al.'s decision to use an off-the-shelf image-segmentation system was pragmatic. The regions it selected are easily visualized and carry some semantic value. However, regions are chosen without any regard to how the classifier makes a decision. To explain a blackbox model, where there is no possible access to the classifier's internal representation, there is likely no better option; any explanation will lack faithfulness.

However, if a user can access the classifier and tailor the explanation system to it, there are ways to choose a more meaningful vocabulary. One interesting method jointly trains a classifier with a natural language, image-captioning system.[13] The classifier uses training data labeled with the objects appearing in the image; the captioning system is labeled with English sentences describing the appearance of the image. By training these systems jointly, the variables in the hidden layers may get aligned to semantically meaningful concepts, even as they are being trained to provide discriminative power. This results in English language descriptions of images that have both high image relevance (from the captioning training data) and high class relevance (from the object recognition training data), as shown in Figure 6.

While this method works well for many examples, some explanations include details that are not actually present in the image; newer approaches, such as phrase-critic methods, may create even better descriptions.[14] Another approach might determine if there are hidden layers in the learned classifier that learn concepts corresponding to something meaningful. For example, Zeiler and Fergus observed that certain layers may function as edge or pattern detectors.[40] Whenever a user can identify the presence of such layers, then it may be preferable to use them in the explanation. Bau et al. describe an automatic mechanism for matching CNN representations with semantically meaningful concepts using a large, labeled corpus of objects, parts, and texture; furthermore, using this alignment, their method quantitatively scores CNN interpretability, potentially suggesting a way to optimize for intelligible models.

However, many obstacles remain. As one example, it is not clear there are satisfying ways to describe important, discriminative features, which are often intangible, for example, textures. An intelligible explanation may need to define new terms or combine language with other modalities, like patches of an image. Another challenge is inducing first-order, relational descriptions, which would enable descriptions such as “a spider because it has eight legs” and “full because all seats are occupied.” While quantified and relational abstractions are very natural for people, progress in statistical-relational learning has been slow and there are many open questions for neuro-symbolic learning.[3]

Facilitating user control with explanatory models. Generating an explanation by mapping an inscrutable model into a simpler, explanatory model is only half of the battle. In addition to answering counterfactuals about the original model, we would ideally be able to map any control actions the user takes in the explanatory model back as adjustments to the original, inscrutable model. For example, as we illustrated how a user could directly edit a GA^{2}M’s shape curve (Figure 4b) to change the model’s response to asthma. Is there a way to interpret such an action, made to an intelligible explanatory model, as a modification to the original, inscrutable model? It seems unlikely that we will discover a general method to do this for arbitrary source models, since the abstraction mapping is not invertible in general. However, there are likely methods for mapping backward to specific classes of source models or for specific types of feature-transform mappings. This is an important area for future study.

Toward Interactive Explanation

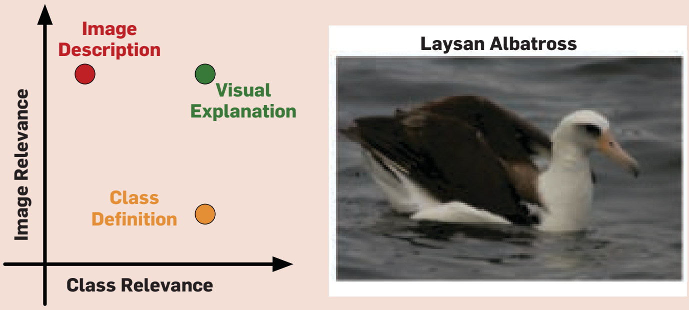
The optimal choice of explanation depends on the audience. Just as a human teacher would explain physics differently to students who know or do not yet know calculus, the technical sophistication and background knowledge of the recipient affects the suitability of a machine-generated explanation. Furthermore, the concerns of a house seeker whose mortgage application was denied due to a FICO score differ from those of a developer or data scientist debugging the system. Therefore, an ideal explainer should model the user’s background over the course of many interactions.

The HCI community has long studied mental models,[31] and many intelligent tutoring systems (ITSs) build explicit models of students’ knowledge and misconceptions.[2] However, the frameworks for these models are typically hand-engineered for each subject domain, so it may be difficult to adapt ITS approaches to a system that aims to explain an arbitrary black-box learner.

Even with an accurate user model, it is likely that an explanation will not answer all of a user’s concerns, because the human may have follow-up questions. We conclude that an explanation system should be interactive, supporting such questions from and actions by the user. This matches results from psychology literature, summarized earlier, and highlights Grice’s maxims, especially those pertaining to quantity and relation. It also builds on Lim and Dey’s work in ubiquitous computing, which investigated the kinds of questions users wished to ask about complex, context-aware applications.[24] We envision an interactive explanation system that supports many different follow-up and

Figure 6. A visual explanation taken from Hendricks et al.[13]

“Visual explanations are both image relevant and class relevant. In contrast, image descriptions are image relevant, but not necessarily class relevant, and class definitions are class relevant but not necessarily image relevant.”



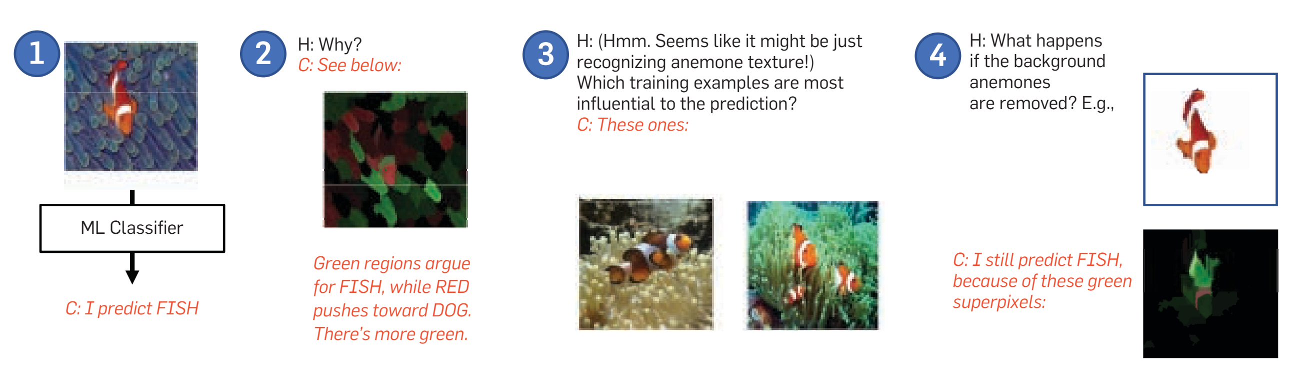
Description: This is a large bird with a white neck and a black back in the water.

Class Definition: The *Laysan Albatross* is a seabird with a hooked yellow beak, black back, and white belly.

Visual Explanation: This is a *Laysan Albatross* because this bird has a hooked yellow beak, white neck, and black back.

Figure 7. An example of an interactive explanatory dialog for gaining insight into a DOG/FISH image classifier.

For illustration, the questions and answers are shown in English language text, but our use of a 'dialog' is for illustration only. An interactive GUI, for example, building on the ideas of Krause et al.,[20] would likely be a better realization.


1
ML Classifier
C: I predict FISH
2
H: Why?
C: See below:
Green regions argue for FISH, while RED pushes toward DOG. There's more green.
3
H: (Hmm. Seems like it might be just recognizing anemone texture!) Which training examples are most influential to the prediction?
C: These ones:
4
H: What happens if the background anemones are removed? E.g.,
C: I still predict FISH, because of these green superpixels:

drill-down actions after presenting a user with an initial explanation:

- *Redirecting the answer by changing the foil.* "Sure, but why didn't you predict class C?"
- *Asking for more detail* (that is, a more complex explanatory model), perhaps while restricting the explanation to a subregion of feature space. "I'm only concerned about women over age 50 ..."
- *Asking for a decision's rationale.* "What made you believe this?" To which the system might respond by displaying the labeled training examples that were most influential in reaching that decision, for example, ones identified by influence functions[19] or nearest neighbor methods.
- *Query the model's sensitivity* by asking what minimal perturbation to certain features would lead to a different output.
- *Changing the vocabulary* by adding (or removing) a feature in the explanatory model, either from a predefined set, by using methods from machine teaching, or with concept activation vectors.[17]
- *Perturbing the input example* to see the effect on both prediction and explanation. In addition to aiding understanding of the model (directly testing a counterfactual), this action enables an affected user who wants to contest the initial prediction: "But officer, one of those prior DUIs was overturned ...?"
- *Adjusting the model.* Based on new understanding, the user may wish to correct the model. Here, we expect to build on tools for interactive machine learning[1] and explanatory debugging,[20,21] which have explored interactions for adding new training examples, correcting erroneous labels in existing data, specifying new features, and modifying shape functions. As mentioned in the previous section, it may be challenging to map user adjustments that are made in reference to an explanatory model, back into the original, inscrutable model.

To make these ideas concrete, Figure 7 presents a possible dialog as a user tries to understand the robustness of a deep neural dog/fish classifier built atop Inception v3.[39] As the figure shows: (1) The computer correctly predicts the image depicts a fish. (2) The user requests an explanation, which is provided using LIME.[33] (3) The user, concerned the classifier is paying more attention to the background than to the fish itself, asks to see the training data that influenced the classifier; the nearest neighbors are computed using influence functions.[19] While there are anemones in those images, it also seems that the system is recognizing a clownfish. (4) To gain confidence, the user edits the input image to remove the background, resubmits it to the classifier and checks the explanation.

Explaining Combinatorial Search

Most of the preceding discussion has focused on intelligible *machine learning*, which is just one type of artificial intelligence. However, the same issues also confront systems based on *deep-lookahead search*. While many planning algorithms have strong theoretical properties, such as soundness, they search over action *models* that include their own assumptions. Furthermore, goal specifications are likewise incomplete.[29] If these unspoken assumptions are incorrect, then a formally correct plan may still be disastrous.

Consider a planning algorithm that has generated a sequence of actions for a remote, mobile robot. If the plan is short with a moderate number of actions, then the problem may be inherently intelligible, and a human could easily spot a problem. However, larger search spaces could be cognitively overwhelming. In these cases, local explanations offer a simplification technique that is helpful, just as it was when explaining machine learning. The vocabulary issue is likewise crucial: how does one succinctly and abstractly summarize a complete search subtree? Depending on the choice of explanatory foil, different answers are appropriate.[8] Sreedharan et al. describe an algorithm for generating the minimal explanation that patches a user's partial understanding of a domain.[37] Work on mixed-initiative planning[7] has demonstrated the importance of supporting interactive dialog with a planning system. Since many AI systems, for example, AlphaGo,[35] combine deep search and machine learning, additional challenges will result from the need to explain interactions between combinatorics and learned models.

Final Thoughts

In order to trust deployed AI systems, we must not only improve their robustness,[5] but also develop ways to make their reasoning intelligible. Intelligibility will help us spot AI that makes mistakes due to distributional drift or incomplete representations of goals and features. Intelligibility will also facilitate control by humans in increasingly common collaborative human/AI teams. Furthermore, intelligibility will help humans learn from AI. Finally, there are legal reasons to want intelligible AI, including the European GDPR and a growing need to assign liability when AI errs.

Depending on the complexity of the models involved, two approaches to enhancing understanding may be appropriate: using an inherently interpretable model, or adopting an inscrutably complex model and generating post hoc explanations by mapping it to a simpler, explanatory model through a combination of currying and local approximation. When learning a model over a medium number of human-interpretable features, one may confidently balance performance and intelligibility with approaches like GA^{2}Ms. However, for problems with thousands or millions of features, performance requirements likely force the adoption of inscrutable methods, such as deep neural networks or boosted decision trees. In these situations, posthoc explanations may be the only way to facilitate human understanding.

Research on explanation algorithms is developing rapidly, with work on both local (instance-specific) explanations and global approximations to the learned model. A key challenge for all these approaches is the construction of an explanation vocabulary, essentially a set of features used in the approximate explanation model. Different explanatory models may be appropriate for different choices of explanatory foil, an aspect deserving more attention from systems builders. While many intelligible models can be directly edited by a user, more research is needed to determine how best to map such actions back to modify an underlying inscrutable model. Results from psychology show that explanation is a social process, best thought of as a conversation. As a result, we advocate increased work on interactive explanation systems that support a wide range of follow-up actions. To spur rapid progress in this important field, we hope to see collaboration between researchers in multiple disciplines.

Acknowledgments. We thank E. Adar, S. Ameshi, R. Calo, R. Caruana, M. Chickering, O. Etzioni, J. Heer, E. Horvitz, T. Hwang, R. Kambhamapti, E. Kamar, S. Kaplan, B. Kim, P. Simard, Mausam, C. Meek, M. Michelson, S. Minton, B. Nushi, G. Ramos, M. Ribeiro, M. Richardson, P. Simard, J. Suh, J. Teevan, T. Wu, and the anonymous reviewers for helpful conversations and comments. This work was supported in part by the Future of Life Institute grant 2015-144577 (5388) with additional support from NSF grant IIS-1420667, ONR grant N00014-15-1-2774, and the WRF/Cable Professorship.

References

1. Amershi, S., Cakmak, M., Knox, W. and Kulesza, T. Power to the people: The role of humans in interactive machine learning. *AI Magazine 35*, 4 (2014), 105–120.
2. Anderson, J.R., Boyle, F. and Reiser, B. Intelligent tutoring systems. *Science 228*, 4698 (1985), 456–462.
3. Besold, T. et al. Neural-Symbolic Learning and Reasoning: A Survey and Interpretation. CoRR abs/1711.03902 (2017). arXiv:1711.03902
4. Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M. and Elhadad, N. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In KDD, 2015.
5. Dietterich, T. Steps towards robust artificial intelligence. *AI Magazine 38*, 3 (2017).
6. Doshi-Velez, F. and Kim, B. Towards a rigorous science of interpretable machine learning. ArXiv (2017), arXiv:1702.08608
7. Ferguson, G. and Allen, J.F. TRIPS: An integrated intelligent problem-solving assistant. In AAAI/IAAI, 1998.
8. Fox, M., Long, D. and Magazzeni, D. Explainable Planning. In IJCAI XAI Workshop, 2017; http://arxiv.org/abs/1709.10256
9. Goodfellow, I.J., Shlens, J. and Szegedy, C. 2014. Explaining and Harnessing Adversarial Examples. ArXiv (2014), arXiv:1412.6572
10. Grice, P. *Logic and Conversation*, 1975, 41–58.
11. Halpern, J. and Pearl, J. Causes and explanations: A structural-model approach. Part I: Causes. *The British J. Philosophy of Science 56*, 4 (2005), 843–887.
12. Hardt, M., Price, E. and Srebro, N. Equality of opportunity in supervised learning. In NIPS, 2016.
13. Hendricks, L., Akata, Z., Rohrbach, M., Donahue, J., Schiele, B. and Darrell, T. Generating visual explanations. In ECCV, 2016.
14. Hendricks, L.A., Hu, R., Darrell, T. and Akata, Z. Grounding visual explanations. ArXiv (2017), arXiv:1711.06465
15. Hilton, D. Conversational processes and causal explanation. *Psychological Bulletin 107*, 1 (1990), 65.
16. Kahneman, D. *Thinking, Fast and Slow.* Farrar, Straus and Giroux, New York, 2011; http://a.co/hGYmXGJ
17. Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F. and Sayres, R. 2017. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors. ArXiv e-prints (Nov. 2017); arXiv:stat.ML/1711.11279
18. Koehler, D.J. Explanation, imagination, and confidence in judgment. *Psychological Bulletin 110*, 3 (1991), 499.
19. Koh, P. and Liang, P. Understanding black-box predictions via influence functions. In ICML, 2017.
20. Krause, J., Dasgupta, A., Swartz, J., Aphinyanaphongs, Y. and Bertini, E. A workflow for visual diagnostics of binary classifiers using instance-level explanations. In IEEE VAST, 2017.
21. Kulesza, T., Burnett, M., Wong, W. and Stumpf, S. Principles of explanatory debugging to personalize interactive machine learning. In IUI, 2015.
22. Lakkaraju, H., Kamar, E., Caruana, R. and Leskovec, J. Interpretable & explorable approximations of black box models. KDD-FATML, 2017.
23. Lewis, D. Causal explanation. *Philosophical Papers 2* (1986), 214–240.
24. Lim, B.Y. and Dey, A.K. Assessing demand for intelligibility in context-aware applications. In *Proceedings of the 11th International Conference on Ubiquitous Computing* (2009). ACM, 195–204.
25. Lipton, Z. The Mythos of Model Interpretability. In *Proceedings of ICML Workshop on Human Interpretability in ML*, 2016.
26. Lombrozo, T. Simplicity and probability in causal explanation. *Cognitive Psychology 55*, 3 (2007), 232–257.
27. Lou, Y., Caruana, R. and Gehrke, J. Intelligible models for classification and regression. In KDD, 2012.
28. Lundberg, S. and Lee, S. A unified approach to interpreting model predictions. NIPS, 2017.
29. McCarthy, J. and Hayes, P. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence* (1969), 463–502.
30. Miller, T. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence 267* (Feb. 2018), 1–38.
31. Norman, D.A. Some observations on mental models. *Mental Models*, Psychology Press, 2014, 15–22.
32. Papadimitriou, A., Symeonidis, P. and Manolopoulos, Y. A generalized taxonomy of explanations styles for traditional and social recommender systems. *Data Mining and Knowledge Discovery 24*, 3 (2012), 555–583.
33. Ribeiro, M., Singh, S. and Guestrin, C. Why should I trust you?: Explaining the predictions of any classifier. In KDD, 2016.
34. Ribeiro, M., Singh, S. and Guestrin, C. Anchors: High-precision model- agnostic explanations. In AAAI, 2018.
35. Silver, D. et al. Mastering the game of Go with deep neural networks and tree search. *Nature 529*, 7587 (2016), 484–489.
36. Sloman, S. Explanatory coherence and the induction of properties. *Thinking & Reasoning 3*, 2 (1997), 81–110.
37. Sreedharan, S., Srivastava, S. and Kambhampati, S. Hierarchical expertise- level modeling for user specific robot-behavior explanations. ArXiv e-prints, (Feb. 2018), arXiv:1802.06895
38. Swartout, W. XPLAIN: A system for creating and explaining expert consulting programs. *Artificial Intelligence 21*, 3 (1983), 285–325.
39. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. Rethinking the inception architecture for computer vision. In CVPR, 2016.
40. Zeiler, M. and Fergus, R. Visualizing and understanding convolutional networks. In ECCV, 2014.

Daniel S. Weld (weld@cs.washington.edu) is Thomas J. Cable/WRF Professor in the Paul G. Allen School of Computer Science & Engineering at the University of Washington, Seattle, WA, USA.

Gagan Bansal (bansalg@cs.washington.edu) is a graduate student in the Paul G. Allen School of Computer Science & Engineering at the University of Washington, Seattle, WA, USA.

Watch the authors discuss this work in the exclusive *Communications* video. https://cacm.acm.org/videos/the-challenge-of-crafting-intelligible-intelligence

ACM Transactions on
Data Science
Association for Computing Machinery

DOI:10.1145/3323921

To view the accompanying paper, visit doi.acm.org/10.1145/3323923 rh

Technical Perspective
Back to the Edge

By Rishiyur S. Nikhil

"YOU MAY FIRE when you are ready, Gridley," is the famous command from Commodore Dewey in the Battle of Manila Bay, 1898. He may not have realized it, but he was articulating the basic principle of dataflow computing, where an instruction can be executed as soon as its inputs are available. Dataflow has long fascinated computer architects as perhaps a more "natural" way for computation circuits to best exploit parallelism for performance.

A visiting alien may be forgiven for experiencing whiplash when shown how we treat parallelism in programs. Mathematical algorithms have abundant parallelism; the only limit is data dependency (an operator can be evaluated when its inputs are available). We code it in a mainstream programming language (C/C++, Python, among others), which has completely sequential semantics (zero parallelism) to make sense of reads and writes to memory. As illustrated in Figure 1, compilers sweat mightily to rediscover some of the lost parallelism in their internal CDFGs (control and data flow graphs), and then produce machine code that, again, is completely sequential. When we execute this on a modern von Neumann CPU, wide-issue, out-of-order circuits once again sweat mightily (burning power) to rediscover parallelism.

The 1970s through early 1990s saw several attempts to avoid these "unnecessary" sequentializations (green circles in Figure 2). Dataflow languages (mostly purely functional) and machine code (dataflow graphs) retained parallelism from the math. Instead of a program counter, each instruction directly named its successor(s) receiving its outputs. Dataflow CPUs directly executed this graph machine code. Nowadays this computation model goes by the acronym EDGE, for explicit dataflow graph execution.

So, why aren't we all using EDGE machines today? A short answer is that they never quite mastered spatial or temporal locality and were subpar on inherently sequential code regions. In contrast, modern von Neumann CPUs excel at this, managing efficient flow of data between circuits that are fast-and-expensive (registers, wires), medium (caches), and slow-and-cheap (DRAMs).

The following paper by Tony Nowatzki, Vinay Gangadhar, and Karthikeyan Sankaralingam describes an innovative approach to exploit both models. From the CDFG, their compiler generates both traditional sequential machine code and a data graph, each being executed on appropriate circuits (blue squares in Figure 2), with efficient hand-off mechanisms. The authors describe extensive studies to validate the viability of this approach for existing codes.

EDGE computing is undergoing a renaissance, with many researchers pursuing related ideas. There are indications that big industry players are also contemplating this direction.[a]

Figure 1. Parallelism during coding, compilation, and execution.

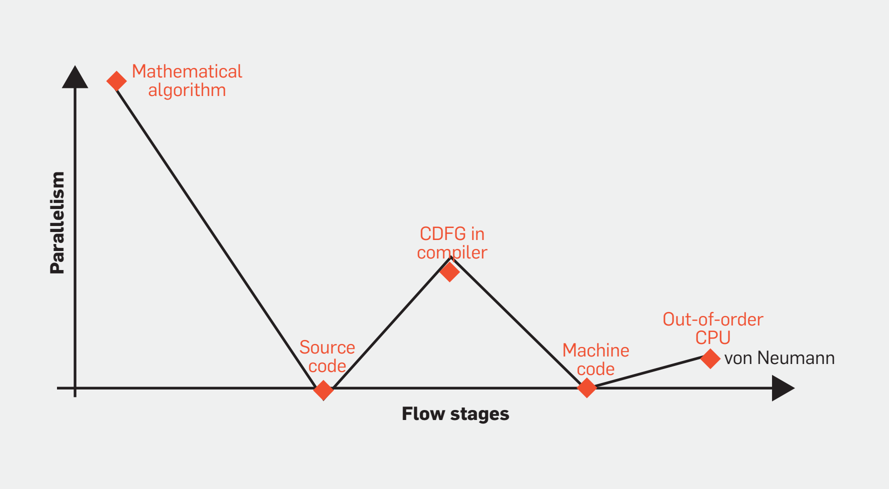
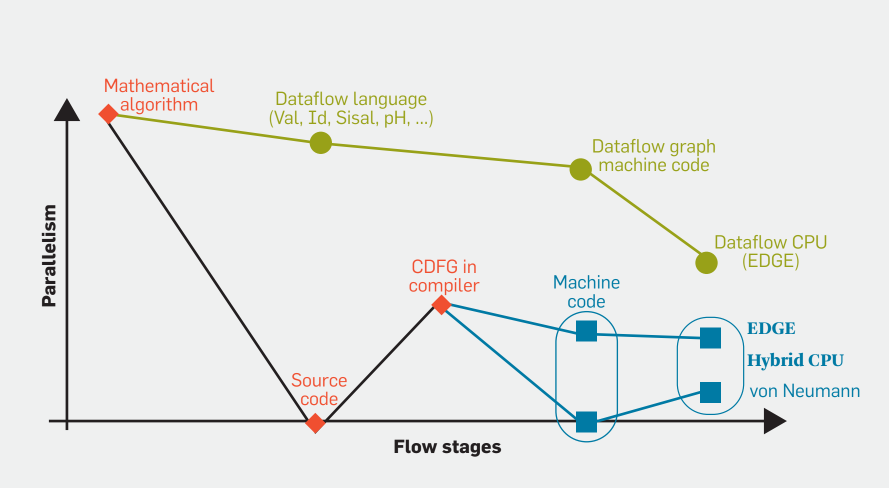


Figure 2. Alternative strategies for exploiting parallelism.



[a] Morgan, T.P. Intel's Exascale dataflow engine drops x86 and von Neumann. *The NEXT Platform*, Aug 30, 2018.

Rishiyur S. Nikhil is Chief Technical Officer at Bluespec, Inc., a semiconductor tool design company in Framingham, MA, USA.

DOI:10.1145/3323923

Heterogeneous Von Neumann/ Dataflow Microprocessors

By Tony Nowatzki, Vinay Gangadhar, and Karthikeyan Sankaralingam

Abstract

General-purpose processors (GPPs), which traditionally rely on a Von Neumann-based execution model, incur burdensome power overheads, largely due to the need to dynamically extract parallelism and maintain precise state. Further, it is extremely difficult to improve their performance without increasing energy usage. Decades-old *explicit-dataflow* architectures eliminate many Von Neumann overheads, but have not been successful as stand-alone alternatives because of poor performance on certain workloads, due to insufficient control speculation and communication overheads.

We observe a synergy between out-of-order (OOO) and explicit-dataflow processors, whereby dynamically switching between them according to the behavior of program phases can greatly improve performance and energy efficiency. This work studies the potential of such a paradigm of heterogeneous execution models, by developing a specialization engine for explicit-dataflow (SEED) and integrating it with a standard out-of-order (OOO) core. When integrated with a dual-issue OOO, it becomes both faster (1.33×) and dramatically more energy efficient (1.70×). Integrated with an in-order core, it becomes faster than even a dual-issue OOO, with twice the energy efficiency.

1. INTRODUCTION

As transistor scaling trends continue to worsen, power limitations make improving the performance and energy efficiency of general purpose processors (GPPs) ever more intractable. The status quo approach of scaling processor structures consumes too much power to be worth the marginal improvements in performance. On top of these challenges, a series of recent microarchitecture level vulnerabilities (Meltdown and Spectre[9]) exploit the underlying techniques which modern processors already rely on for exploiting instruction-level parallelism (ILP).

Fundamental to these issues is the Von Neumann execution model adopted by modern GPPs. To make the contract between the program and the hardware simple, a Von Neumann machine logically executes instructions in the order specified by the program, and dependences are implicit through the names of storage locations (registers and memory addresses). However, this has the consequence that exploiting ILP effectively requires sophisticated techniques. Specifically, it requires (1) dynamic discovery of register/memory dependences, (2) speculative execution past unresolved control flow instructions, and (3) maintenance of the precise program state at each dynamic instruction should it be need to be recovered (e.g., an exception due to a context switch).

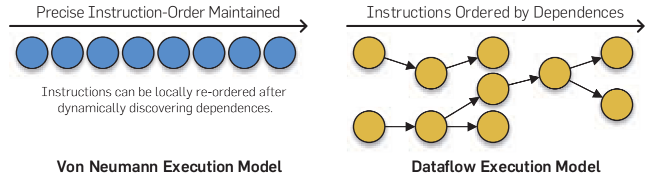
The above techniques are the heart of modern Von Neumann out-of-order (OOO) processors, and each technique requires significant hardware overhead (register renaming, instruction wakeup, reorder-buffer maintenance, speculation recovery, etc.). In addition, the instruction-by-instruction execution incurs considerable energy overheads in pipeline processing (fetch, decode, commit, etc.). As for security, the class of vulnerabilities known as Meltdown and Spectre all make use of speculative execution of one form or another, adding another reason to find an alternative.

Interestingly, there exists a well-known class of architectures that mitigate much of the above called *explicit-dataflow* (e.g., Tagged Token Dataflow,[1] TRIPS,[3] WaveScalar[20]). Figure 1 shows that the defining characteristic of this execution model is how it encodes both control and data dependences explicitly, and the dynamic instructions are ordered by these dependences rather than a total order. Thus, a precise program state is not maintained at every instruction. The benefit is extremely cheap exploitation of instruction-level parallelism in hardware, because no dynamic dependence construction is required.

However, explicit-dataflow architectures show no signs of replacing conventional GPPs for at least three reasons. First, control speculation is limited by the difficultly of implementing efficient dataflow-based squashing. Second, the latency cost of explicit data communication can be prohibitive.[2] Third, compilation challenges for general workloads have proven hard to surmount.[5] Although a dataflow-based execution model may help many workloads, it can also significantly hamper others.

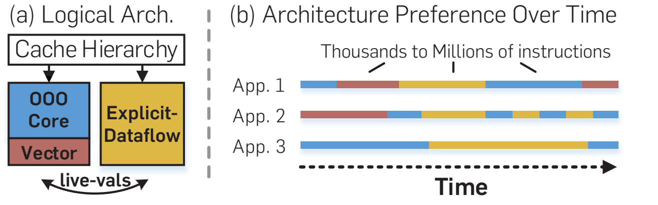
Unexplored opportunity: What is unexplored so far is the fine-grained interleaving of explicit-dataflow with Von Neumann execution—that is, the theoretical and practical limits of being able to switch with low cost between an explicit-dataflow hardware/ISA and a Von Neumann ISA. Figure 2(a) shows a logical view of such a heterogeneous architecture, and Figure 2(b) shows the capability of this architecture to exploit fine-grain (thousands to millions of instructions) application phases. This is interesting now, as

Figure 1. Von Neumann vs. dataflow at a glance.


Precise Instruction-Order Maintained
Instructions can be locally re-ordered after dynamically discovering dependences.
Von Neumann Execution Model
Instructions Ordered by Dependences
Dataflow Execution Model

The original version of this paper is entitled "Exploring the Potential of Heterogeneous Von Neumann/Dataflow Execution Models" and was published in ISCA 2015.

Figure 2. Taking advantage of dynamic program behavior.



trends mean that on-chip power is more limited than area; this creates "dark-silicon," portions of the chip that cannot be kept active due to power constraints. The two major implications are that energy efficiency is the key to improving scalable performance, and that it becomes rationale to add specialized hardware which is only in-use when profitable.

With such a hardware organization, many open questions arise: Are the benefits of fine-grained interleaving of execution models significant enough? How might one build a practical and small footprint dataflow engine capable of serving as an offload engine? Which types of GPP cores can get substantial benefits? Why are certain program region-types suitable for explicit-dataflow execution?

To answer these questions we make the following contributions. Most importantly, we identify (and quantify) the potential of switching between OOO and explicit-dataflow at a fine grain. Next, we develop a specialization engine for explicit-dataflow (SEED) by combining known dataflow-architecture techniques, and specializing the design for program characteristics where explicit-dataflow excels as well as simplifying and common program structures (loops/nested loops). We evaluate the benefits through a design-space exploration, integrating SEED into little (in-order), medium (OOO2), and big (OOO4) cores. Our results demonstrate large energy benefits over >1.5×, and speedups of 1.67×, 1.33×, and 1.14× across little, medium, and big cores. Finally, our analysis illuminates the relationship between workload properties and dataflow profitability: code with high memory parallelism, instruction parallelism, and control noncriticality is highly profitable for dataflow execution. These are common properties for many emerging workloads in machine learning and data processing.

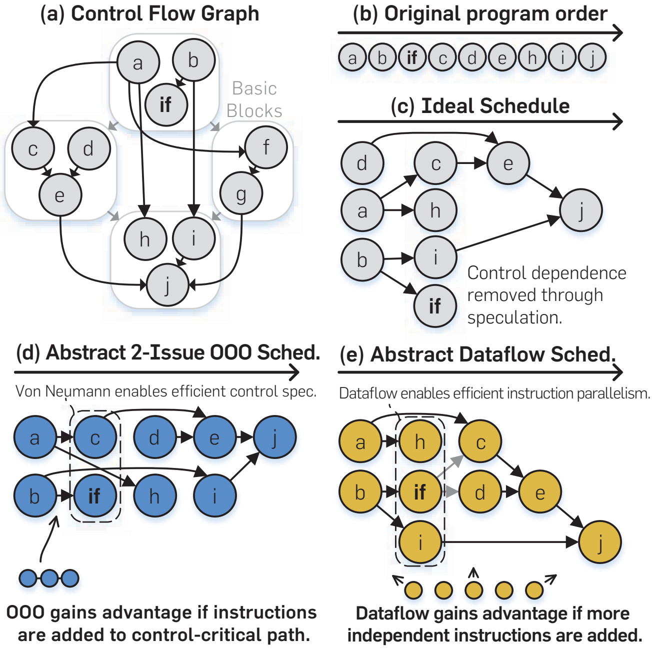
2. UNDERSTANDING VON NEUMANN/DATAFLOW SYNERGY

Understanding the trade-offs between a Von Neumann machine, which reorders instructions implicitly, and a dataflow machine, which executes instructions in dependence order, can be subtle. Yet, the trade-offs have profound implications. We attempt to distill the intuition and quantitative potential of a heterogeneous core as follows.

2.1. Intuition for execution model affinity

The intuitive trade-off between the two execution models is that explicit-dataflow is more easily specializable for high issue width and instruction window size (due to lack of need to discover dependences dynamically), whereas an implicit-dataflow architecture is more easily specializable for speculation (due to its maintenance of precise state of all dynamic instructions in total program order).

Figure 3. Von Neumann and dataflow execution models.


(a) Control Flow Graph
a
b
if
Basic Blocks
c
d
e
f
g
h
i
j
(b) Original program order
a b if c d e h i j
(c) Ideal Schedule
d c e a h j b i if
Control dependence removed through speculation.
(d) Abstract 2-Issue OOO Sched.
Von Neumann enables efficient control spec.
a c d e j
b if h i
(e) Abstract Dataflow Sched.
Dataflow enables efficient instruction parallelism.
a h c
b if d e
i j

OOO gains advantage if instructions are added to control-critical path.

Dataflow gains advantage if more independent instructions are added.

The performance implications can be seen in an example in Figure 3(a), which has a single control decision labeled as ⓘⓕ. In (b), we show the program instruction order for one iteration of this code, assuming the left branch was taken. Figure 3(c) shows the ideal schedule of these instructions on an ideal machine (one instruction per cycle). The key to the ideal execution is both the reordering of dependent instructions (ⓒ, ⓓ) before the control decision is resolved, as well as being able to execute many instructions in parallel.

A Von Neumann OOO machine has the advantage of speculative execution, but the disadvantage is the complexity of implementing hardware for issuing multiple instructions per cycle (issue width) when the dependences are determined dynamically. Therefore, (d) shows how a dual-issue OOO takes five cycles because there was not enough issue bandwidth for both ⓓ and ⓗ before the third cycle.

A dataflow processor can easily be designed for high issue width due to dependences being explicitly encoded into the program representation. However, we assume here that the dataflow processor does not perform speculation, because of the difficulty of recovering when a precise order is not maintained. Therefore, in Figure 3(e), the dataflow processor's schedule, ⓒ and ⓓ; must execute after the ⓘⓕ.

Although the example suggests the benefits of control specialization and wide issue widths are similar, in practice, the differences can be stark, which we can demonstrate with slight modifications to the example. If we add several instructions to the critical path of the control decision (between ⓑ and ⓘⓕ), the OOO core can hide these through control speculation. If instead we add more parallel instructions, the explicit-dataflow processor can execute these in parallel, whereas these may be serialized in the OOO Von Neumann machine. Explicit-dataflow can also be beneficial if the ⓘⓕ is unpredictable, and the OOO is anyway serialized.

2.2. Quantitative potential

A natural next question is how much potential benefit could a heterogeneous Von Neumann/dataflow core provide. The potential benefits of an *ideal* hybrid architecture (ideal dataflow + four-wide OOO) relative to a standard OOO core are as shown in Figure 4(a), which where each speedup bar is labeled with the percentage of execution time that dataflow execution is profitable. Figure 4(b) shows the overall energy and performance trends for three different GPPs.

These results indicate that dataflow specialization has significant potential, up to 1.5× performance for an OOO4 GPP (2× for OOO2), as well as over 2× average energy-efficiency improvement. Furthermore, the preference for explicit-dataflow is frequent, covering around 65% of execution time, but also intermittent and application-phase dependent. The percentage of execution time in dataflow mode varies greatly, often between 20% and 80%, suggesting that phase types can exist at a fine grain inside an application.

Overall, this suggests that a heterogeneous Von Neumann/explicit-dataflow architecture with fine-granularity switching can provide significant performance improvements along with power reduction, and thus lower energy.

Remaining challenge: Although many high-performance explicit-dataflow architectures have been proposed over the last several decades, the remaining challenge is how to achieve the same benefits while avoiding a more heavyweight general-purpose explicit-dataflow engine (for example, WaveScalar[20] or TRIPS,[3] see Figure 5). The approach we will take is to combine known dataflow mechanisms, while simplifying and specializing for the common workload characteristics where dataflow excels.

3. SEED: AN ARCHITECTURE FOR FINE-GRAIN DATAFLOW SPECIALIZATION

Based on our previous analysis and insights, there are three primary requirements for a dataflow specialization engine: (1) low area and power, so integration with the GPP is feasible; (2) enough generality to target a wide variety of workloads; and (3) achieving the benefits of dataflow execution with few overheads.

The dataflow processor is only constrained by the program's control and data-dependencies, but retains the same memory system. Note that *no* nonlocal program modifications are performed (no loop reordering/tiling/layout-transforms/etc.). It is also nonspeculative and incurs latency when transferring values between control regions. For energy, only functional units and caches are considered.

First, we propose that requirement 1, low area and power, can be addressed by focusing on a common, yet simplifying case: *fully-inlined loops and nested loops* with a limited total static instruction count. This helps limit the size of the dataflow tags and eliminates the need for an instruction cache; both of which reduce hardware complexity. In addition, ignoring recursive regions and only allowing in-flight instructions from a single context eliminates the need for tag matching hardware. Targeting nested-loops also satisfies requirement 2: these regions can cover a majority of real applications' dynamic instructions.

For low-overhead dataflow execution, requirement 3, communication must be lowered while maintaining parallelism. For this, we first use a *distributed-issue* architecture, which enables high-instruction throughput with *low-ported RAM* structures. Second, we use a *multibus network* for sustaining instruction communication throughput at low latency. Third, we use *compound instructions* to reduce communication overhead. The proposed design is SEED: specialization engine for explicit-dataflow, shown at a high level in Figure 6, and explained next.

Figure 4. Ideal dataflow specialization potential.

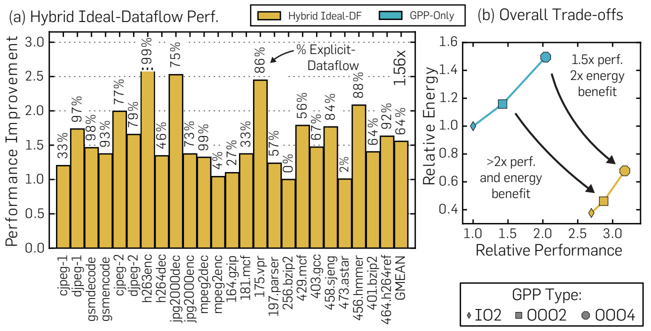


Figure 5. Relationship to dataflow architectures.

Are prior dataflow architectures sufficient?
Two reasons motivate innovation beyond existing dataflow architectures for the heterogeneous core. First, most prior dataflow architectures have significant area and power overheads, because they are targeted at whole-program execution and must handle arbitrary code. For example, TRIPS[3] uses a dynamically routed mesh network to exploit many different forms of parallelism. WaveScalar[20] uses large hierarchical interconnects and complex tag-matching, in part because it needs to disambiguate instructions from multiple function contexts. Second, their designs do not consider the costs of switching at low-overhead (for example, not relying on prediction state that requires warm-up). On the other hand, existing in-core accelerators that act as offload engines have much lower power and area, but are not general enough. None of them can offload entire loop regions in general—only the com- putation in CCA[4] and DySER[6] or hot loop-traces in BERET.[7]

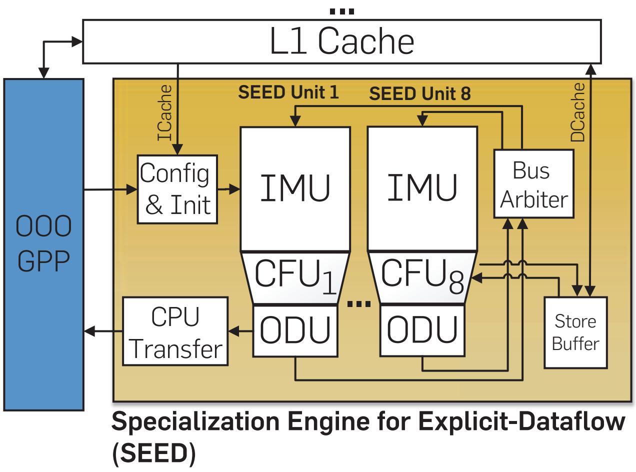
How are prior dataflow techniques used?
SEED is highly inspired by previous decades of dataflow research. For example, Monsoon[19] improves the efficiency of matching operands using an Explicit Token Store, a concept we borrow for SEED's explicit operand buffer. The mechanisms that SEED uses for efficient and general dataflow-based control are derived from WaveScalar,[20] the concept of a copro- cessor,[12] and the concept of efficient compound FUs from BERET.[7]

3.1. Von Neumann core integration

Adaptive execution: To adaptively apply explicit-dataflow specialization, we use a technique similar to bigLITTLE, except that we restrict the entry points of specializable regions to fully-inlined loops or nested loops. This simplifies integration with a different ISA. Targeting longer nested-loop regions reduces the cost of configuration and GPP core synchronization.

GPP integration: SEED uses the same cache hierarchy as the GPP, which facilitates fast switching (no data-copying through memory) and preserves cache coherence. SEED adds architectural state, which must be maintained at context switches. Lastly, functional units (FUs) could be shared with the GPP to save area (by adding bypass paths); this work considers stand-alone FUs.

Figure 6. High-level SEED integration and organization (IMU: instruction management unit; CFU: compound functional unit; ODU: output distribution unit).



3.2. Dataflow execution model

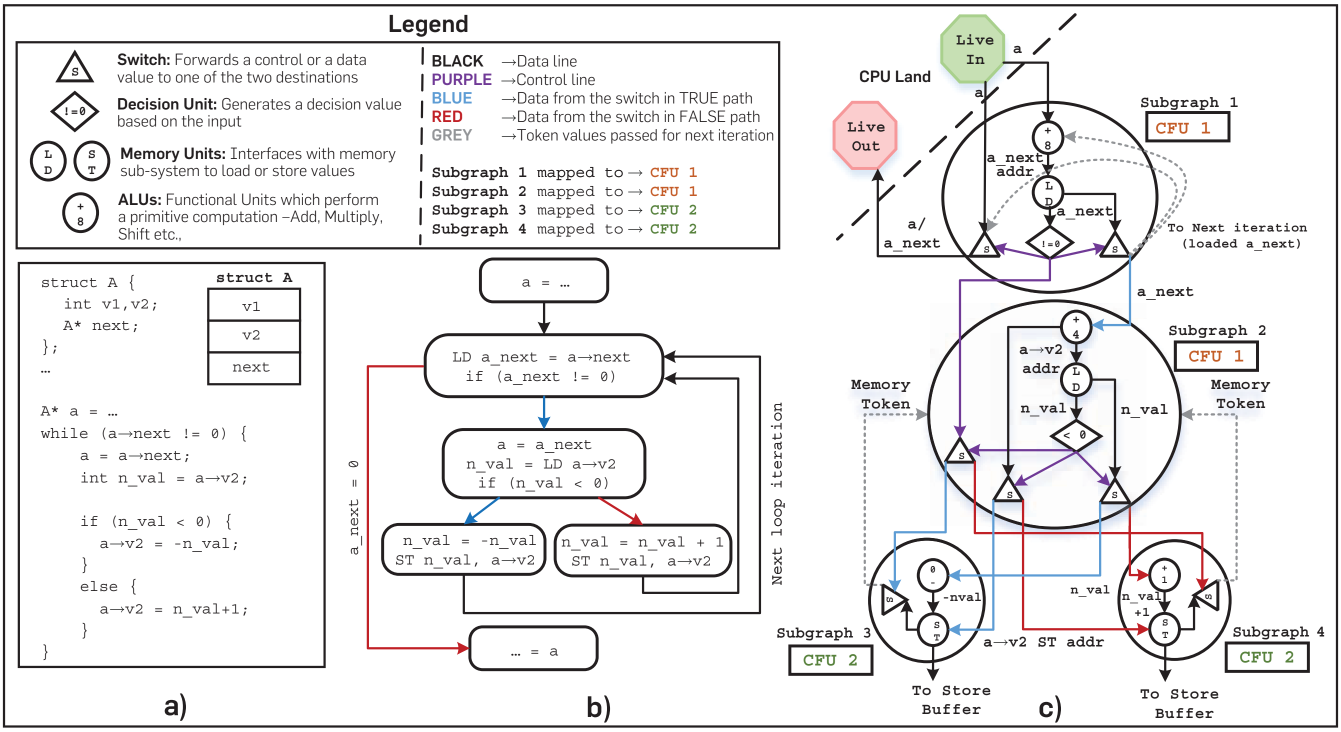
SEED's execution model closely resembles prior dataflow architectures, but is restricted for loops/nested-loops, and adds the use of compound instructions.

We use a running example to aid explanation: a simple linked-list traversal where a conditional computation is performed at each node. Figure 7(a) shows the original program, (b) the Von Neumann control flow graph (CFG) representation, and (c) SEED's explicit-dataflow representation.

Data-dependence: Similar to other dataflow representations, SEED programs follow the dataflow firing rule: instructions execute when their operands are ready. To initiate computation, live-in values are sent from the host. During dataflow execution, each instruction forwards its outputs to dependent instructions, either in the same iteration (solid line in Figure 7(c)), or in a subsequent iteration (dotted line). For example, the `a_next` value loaded from memory is passed on to the next iteration for address computation.

Control-flow strategy: Control dependencies between instructions are converted into data dependencies. SEED uses a *switch* instruction, which forwards values to one of two possible destinations depending on the input control signal. In the example, depending on the `n_val` comparison, `v2` is forwarded to either the `if` or `else` branch. This strategy enables control-equivalent regions to execute in parallel.

Figure 7. (a) Example C loop; (b) control flow graph (CFG); (c) SEED program representation.


Legend
S
Switch: Forwards a control or a data value to one of the two destinations
!=0
Decision Unit: Generates a decision value based on the input
LD
ST
Memory Units: Interfaces with memory sub-system to load or store values
+8
ALUs: Functional Units which perform a primitive computation –Add, Multiply, Shift etc.,
BLACK →Data line
PURPLE →Control line
BLUE →Data from the switch in TRUE path
RED →Data from the switch in FALSE path
GREY →Token values passed for next iteration
Subgraph 1 mapped to → CFU 1
Subgraph 2 mapped to → CFU 1
Subgraph 3 mapped to → CFU 2
Subgraph 4 mapped to → CFU 2
struct A
v1
v2
next
a = ...
LD a_next = a→next
if (a_next != 0)
a = a_next
n_val = LD a→v2
if (n_val < 0)
n_val = -n_val
ST n_val, a→v2
n_val = n_val + 1
ST n_val, a→v2
... = a
a_next = 0
Next loop iteration
Live In
a
CPU Land
a
Live Out
a/
a_next
Subgraph 1
CFU 1
+8
a_next
addr
LD
a_next
S
!=0
S
To Next iteration
(loaded a_next)
a_next
Subgraph 2
CFU 1
+4
a→v2
addr
LD
Memory Token
Memory Token
n_val
n_val
< 0
S
S
S
0
-
-nval
S
ST
n_val
+
1
n_val
+1
S
ST
Subgraph 3
CFU 2
a→v2 ST addr
Subgraph 4
CFU 2
To Store Buffer
To Store Buffer
a)
b)
c)

```
struct A {
  int v1,v2;
  A* next;
};
...

A* a = ...
while (a→next != 0) {
    a = a→next;
    int n_val = a→v2;

    if (n_val < 0) {
      a→v2 = -n_val;
    }
    else {
      a→v2 = n_val+1;
    }
}
```

Enforcing memory-ordering: SEED uses a primarily software approach to enforce memory-ordering. When the compiler identifies dependent (or aliasing) instructions, the program serializes these through explicit tokens. In this example, the stores of `n_val` can conflict with the load from the next iteration (e.g., when the linked list contains a loop), and therefore, memory dependence edges are added.

Executing compound instructions: To mitigate communication overheads, the compiler groups primitive instructions (e.g., adds, shifts, switches, etc.) into subgraphs and executes them on compound functional units (CFUs). These are logically executed atomically. The example program contains four subgraphs, mapped to two CFUs.

3.3. SEED microarchitecture

SEED achieves high instruction parallelism and simplicity by using eight distributed computation units. Each of these *SEED units* is organized around one CFU, and units communicate together over a network, as shown in Figure 6.

Compound functional unit (CFU): CFUs are composed of a fixed network of primitive FUs (adders, multipliers, logical units, switch units, etc.), where unused portions of the CFU are bypassed when not in use. Long latency instructions (e.g., loads) can be buffered and passed by subsequent instructions. Our design uses the CFU mix from existing work,[7] where CFUs contain 2–5 operations. CFUs which have memory units will issue load and store requests to the host's memory management unit. Load requests access a store buffer for store-to-load forwarding.

Instruction management unit (IMU): The IMU has three responsibilities. First, it stores up to 32 compound instructions, each with a maximum of four operands each for up to four dynamic loop iterations (equivalent to a 1024-entry instruction window). Second, it selects instructions with ready operands for execution on the CFU, giving priority to the oldest instruction. Third, the IMU routes incoming values from the network to appropriate storage locations based on the incoming instruction tag.

Communication: The ODU is responsible for distributing the output values and destination packets (SEED unit + instruction location + iteration offset), to the bus network, and buffering them during bus conflicts. A bus interconnect forwards output packets from the ODU to SEED unit IMU's which use the corresponding operands. Therefore, dependent instructions communicating over the bus cannot execute in back-to-back cycles, a limitation of distributed dataflow.

4. SEED COMPILER DESIGN

The two main responsibilities of the compiler are determining which regions to specialize and scheduling instructions into CFUs inside SEED regions.

Region selection: The compiler must find or create fully-inlined nested-loop regions, which are small enough to match SEED's operand/instruction storage. Also, the inner loop should be unrolled for instruction parallelism. An Amdahl-tree based approach can be used to select regions.[16] Also, we should avoid regions where the OOO core (through control speculation) or the SIMD units would have performed better. One approach is to use simple heuristics, for example, avoid control-critical regions. A dynamic approach can be more flexible; for example, training online predictors to give a runtime performance estimate based on per-region statistics. Related work explores this in detail,[16, 18] and this work simply uses a static oracle scheduler (see Section 5).

Instruction scheduling: The instruction scheduler forms compound instructions and assigns them to units. Its job is to balance communication cost by creating large compound instructions, while also ensuring that combining instructions does not artificially increase the critical path length. To solve this, we use integer linear programming, specifically extending a general scheduling framework for spatial architectures[17] with the ability to model instruction bundling.

5. EVALUATION METHODOLOGY

For evaluating SEED, OOO core specialization techniques, and the other designs we compare to, we employ a TDG-based modeling methodology.[15] We use Mc-PAT[11] with 22nm technology to estimate power and area. Von Neumann core configurations are given in Table 1.

The benchmarks we chose were from SPECint and Mediabench,[10] representing a variety of control and memory irregularity, as well as some regular benchmarks. To eliminate compiler/runtime heuristics on when to use which architecture, we use an oracle scheduler, which uses previous runs to decide when to use the OOO core, SEED, or SIMD.

6. EVALUATING DATAFLOW SPECIALIZATION POTENTIAL

To understand the potentials and trade-offs of dataflow specialization, we explore the prevalence of required program structure, per-region performance, and overall heterogeneous core benefits.

6.1. Program structure

Nested loop prevalence: Figure 8 shows cumulative distributions of dynamic instruction coverage with varying dynamic region granularity, assuming maximum 1024 instructions. Considering regions with a duration of 8K dynamic instructions or longer (x-axis), nested loops can cover 60% of total instructions, whereas inner loops cover only 20%. Nested loops also greatly increase the region duration for a given percentage coverage (1K–64K for 40% coverage).

Compound instruction prevalence: Figure 9 is a histogram of per-benchmark compound instruction sizes which the compiler created, showing on average 2–3 instructions. This

Table 1. Von Neuman core configurations.

GPP	Characteristics
Little (IO2)	Dual issue, 1 load/store port.
Medium (OOO2)	64 entry ROB, 32 entry IW, LSQ: 16 ld/20 st, 1 ld/st ports, speculative scheduling.
Big (OOO4)	168 entry ROB, 48 entry IW, LSQ: 64 ld/36 st, 2 ld/st ports, speculative scheduling.
Common	x86 ISA, 256-bit SIMD, 2-way 32KiB I$, 64KiB L1D$ (4 cycle latency), 8-way 2MB L2$ (22 cycle hit latency), 2GHz.

Figure 8. Cumulative % contribution for decreasing dynamic region lengths. Static region size £ 1024 insts.

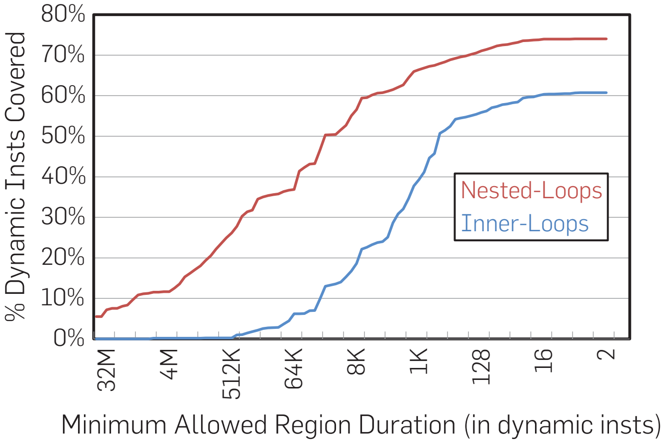


Figure 9. Compound instruction size histogram.

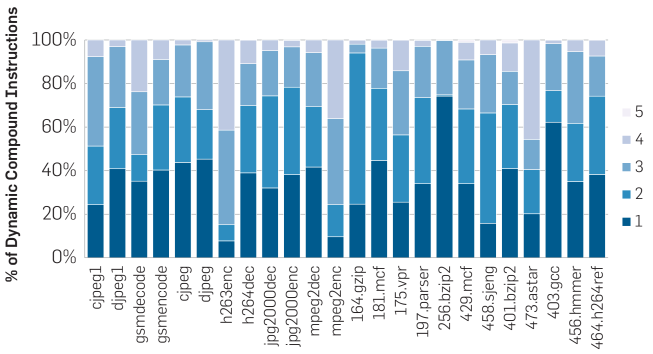
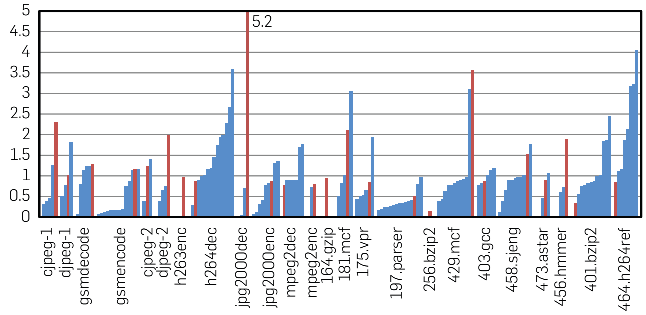


Figure 10. Per-region SEED speedups. Highest-contributing region shown in red.



is relatively high considering that compound instructions cannot cross control regions. Some singletons are necessary; however, either because control regions lack dependent computation, or because combining certain instructions would create additional critical-path dependencies.

6.2. Per-region performance analysis

First, we compare the speedups of SEED to our most aggressive design (OOO4) on a per-region basis. Figure 10 shows SEED's speedup for the recurring nested-loop program regions (each >1% total insts), where the region with the highest contribution to the execution time of the original program is shown in red. Overall, speedup varies dramatically due to the significant differences in program characteristics. Around 3×–5× speedup is possible, and many regions show significant speedup. We next examine the reasons for performance differences of the highest-contribution regions in several categories, as follows.

Performance and energy benefit regions: Compared to the OOO4-wide core, SEED can provide high speedups by exploiting ILP in compute-intensive regions (e.g., `jpg2000dec`, `cjpeg`, and `djpeg`) and from using an effectively larger instruction window for achieving higher memory parallelism (e.g., `181.mcf` and `429.mcf`). The latter have high cache miss rates and clog the instruction window of the OOO processor. Across these regions, indirect memory access is common, which precludes SIMD vectorization.

Energy benefit-only regions: These regions have similar performance to the OOO4, but are more energy efficient by 2×–3×. Here, ILP tends to be lower, but control is mostly off the critical path, allowing dataflow to compete (e.g., `djpeg-1` and `h264dec`). Although `gsmencode` and `164.gzip` actually have high potential ILP, they are burdened by communication between SEED units. Contrastingly, `473.astar` and `jpg2000enc` have significant control, but still perform close to the OOO core. These benchmarks make up for the lack of speculation by avoiding branch mispredictions and relying on the dataflow-based control.

Performance loss regions: The most common reason for performance loss is communication latency on the critical path (e.g., `403.gcc`, `mpeg2dec`, and `mpeg2enc`), as well as predictable data-dependent control (e.g., `401.bzip2`). These are fundamental dataflow limitations. In two cases, configuration overhead was burdensome (`464.h264ref` and `197.parser`). Finally, some of these regions are vectorized on the GPP, and SEED is not optimized to exploit data-parallelism. In practice, the above regions would not be executed on SEED.

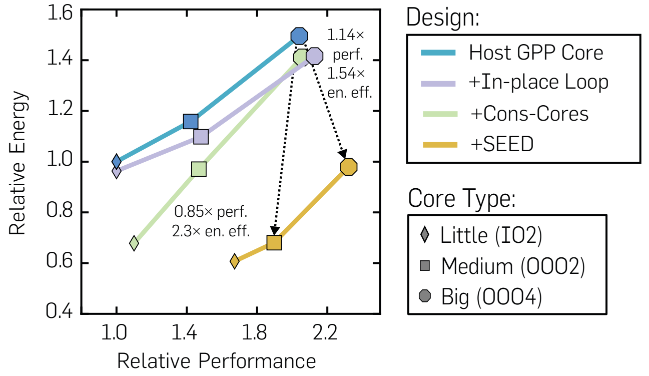
In summary, speedups come from exploiting higher memory parallelism and instruction parallelism, and avoiding misspeculation on unpredictable branches. Slowdowns come from the extra latency cost on more serialized computations.

6.3. Overall performance/energy trade-offs

Finally, we consider the overall performance when integrated with a little, medium, and big core, and compare explicit-dataflow specialization with existing techniques for targeting irregular codes. In-place loop execution, similar to Revolver,[8] locks looping instructions into the instruction window to prevent redundant pipeline overheads such as fetch/decode/execute, but does not otherwise change the OOO execution model. Conservation cores[21] use software-defined region-specific accelerators, but they do not exploit dataflow-based control (only in-order control and memory). Figure 11 shows the relative performance and energy benefits, normalized to the in-order core alone.

SEED improves performance and energy efficiency across GPP cores types, significantly more than existing accelerator

Figure 11. Overall performance and energy benefit.



and microarchitectural approaches. For the little, medium, and big cores, SEED provides 1.65×, 1.33×, and 1.14× speedup, and 1.64×, 1.7×, and 1.53× energy efficiency, respectively. The energy benefits come primarily from the prevalence of regions where dataflow execution can match the host core's performance; this occurs 71%, 64%, and 42% of the time, for the little, medium, and big Von Neumann cores, respectively.

Understanding disruptive trade-offs: Perhaps more interesting is the disruptive changes that explicit-dataflow specialization introduces for computer architects. First, the OOO2+SEED is actually reasonably close in performance to an OOO4 processor on average, within 15%, while reducing energy 2.3×. Additionally, our estimates suggest that an OOO2+SEED occupies less area than an OOO4 GPP core. Therefore, a hybrid dataflow system introduces an interesting path toward a high-performance, low-energy microprocessor: start with an easier-to-engineer modest OOO core, and add a simple, nongeneral-purpose dataflow engine.

An equally interesting trade-off is to add a hybrid dataflow unit to a larger OOO core—SEED+OOO4 has much higher energy efficiency (1.54×) with additional performance improvements of 1.14×. This is a significant leap for energy-efficiency, especially considering the difficulty of improving the efficiency for complex, irregular workloads such as SpecINT.

Overall, all cores can achieve significant energy benefits, little and medium cores can achieve significant speedup, and big cores receive modest performance improvement.

7. DISCUSSION

Dataflow specialization is a broadly applicable principle for both general-purpose processors and accelerators. We outline our view on the potentially disruptive implications in these areas as well as potential future directions.

7.1. General purpose cores

In this work, we showed how a dataflow processor can more efficiently take over and execute certain phases of application workloads, based on their properties. This can be viewed visually, as shown in Figure 12, where we show architecture affinity for programs along dimensions of control and memory regularity. Figure 12(a) shows how prior programmable specialization techniques only focus on a narrow range of workloads—for example, SIMD can speedup highly regular program phases ❶ only.

Figure 12(b) shows how dataflow specialization further cuts into the space of programs that traditional architectures are best at. Specifically, when the OOO processor's issue width and instruction window size limits the achievable ILP (region ❸), explicit-dataflow processors can exploit this through distributed dataflow, as well as more efficient execution under control unpredictability (region ❹). Beyond these region types, dataflow specialization can be applied to create engines that target other behaviors, such as repeatable control ❺, or to further improve highly regular regions by combining dataflow with vector-communication ❶.

Future directions: The disruptive potential of exploiting common program phase behavior using a heterogeneous dataflow execution model can have significant implications leading to several important directions:

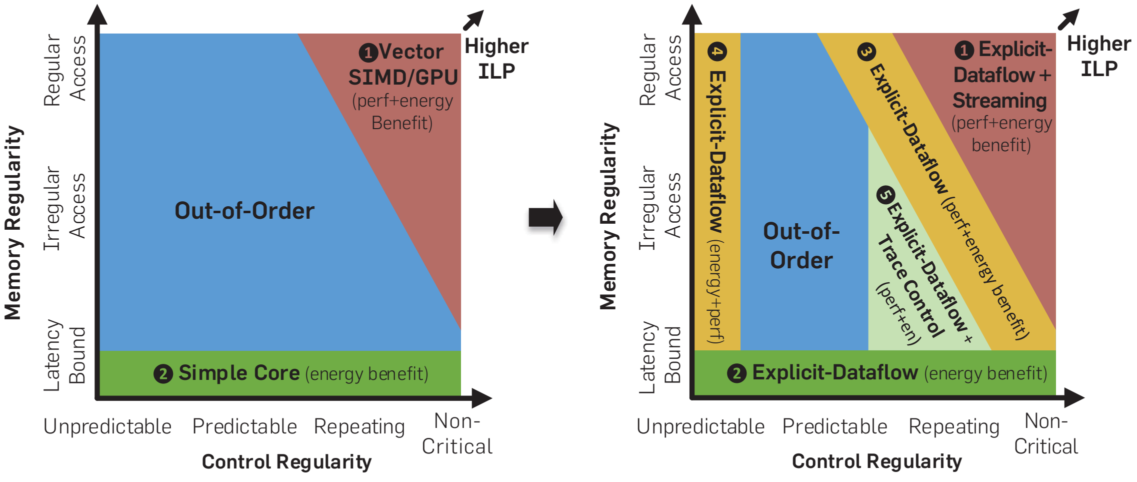
- **Reduced importance of aggressive out-of-order:** Dataflow engines which can exploit high ILP phases can reduce the need for aggressive and power-inefficient out-of-order cores. As a corollary, the design of modest-complexity loosely coupled cores should in principle be less design effort than a complex OOO core. This could lower the cost-of-entry into the general-purpose core market, increasing competition and spurring innovation.
- **Radical departure from status quo:** The simple and modular integration of engines targeting different behaviors, combined with microarchitecture-level dynamic compilation for dataflow ISAs[22] can enable such designs to be practical. This opens the potential of exploring designs with radically different microarchitectures and software interfaces, ultimately opening a larger and more exciting design space.
- **An alternative secure processor:** An open question is how to build future secure processors that are immune to attacks such as Meltdown and Spectre.[9] One approach is to simply avoid speculation; this work shows that an in-order core plus SEED may only lose on average around 20% performance with respect to an OOO core alone, at much lower energy.

7.2. Accelerators

In contrast to general-purpose processors, accelerators are purpose-built chips integrated at a coarse grain with computing systems, for workloads important-enough to the market to justify their design and manufacturing cost. A persistent challenge facing accelerator design is that in order to achieve desired performance and energy efficiency, accelerators often sacrifice generality and programmability, using application or domain-specific software interfaces. Their architecture and microarchitecture is narrowly tailored to the particular domain and problem being solved.

The principle of heterogeneous Von Neumann/dataflow architectures can help to create a highly efficient accelerator without having to give up on domain-generality. Inspired by the insights here, we demonstrated that domain-specific accelerators rely on fundamentally common specialization principles: specialization of computation, communication,

Figure 12. Program phase affinity by application characteristics. Memory ranges from *regular* and data-independent, to *irregular* and data-dependent but with parallelism, to *latency bound* with no parallelism. Control can range from *noncritical* or not present, critical but *repeating*, not repeating but *predictable*, to *unpredictable* and data-dependent.



(a) Prior Specialization Techniques

(b) Enabled Specialization Techniques

concurrency, data-reuse, and coordination.[14] A dataflow model of computation is especially suitable for exploiting the first three principles for massive parallel computation, whereas a Von Neumann model excels at the coordination of control decisions and ordering. We further addressed programmable specialization by proposing a Von Neumann/dataflow architecture called stream-dataflow,[13] which specifies memory access and communication as streams, enabling effective specialization of data-reuse in caches and scratchpad memories.

Future directions: The promise of dataflow specialization in the accelerator context is to enable freedom from application-specific hardware development, leading to two important future directions.

- **An accelerator architecture:** The high energy and area-efficiency of a Von Neumann/dataflow accelerator, coupled with a well-defined hardware/software interface, enables the almost paradoxical concept of an accelerator *architecture*. We envision that a dataflow-specialized ISA such as stream-dataflow, along with essential hardware specialization principles, can serve as the basis for future innovation for specialization architectures. Its high efficiency makes it an excellent baseline comparison design for new accelerators, and the ease of modifying its hardware/software interface can enable integration of novel forms of computation and memory specialization for challenging workload domains.
- **Compilation:** How a given program leverages Von Neumann and dataflow mechanisms can have tremendous influence on attainable efficiency, and some methodology is required to navigate this design space. The fundamental compiler problem remains extracting and expressing parallelism and locality. The execution model and application domains make these problems easier to address. Applications for which accelerators are amenable are generally well-behaved (keeping to a minimum or avoiding pointers, etc.). The execution model and architecture provides interfaces to cleanly expose the application's parallelism and locality to the hardware. This opens up exciting opportunities in compiler and programming languages research to target accelerators.

8. CONCLUSION

This article observed a synergy between Von Neumann and dataflow processors due to variance in program behaviors at a fine grain and used this insight to build a practical processor, SEED. It enables potentially disruptive performance and energy efficiency trade-offs for general-purpose processors, pushing the boundary of what is possible given only a modestly complex core. This approach of specializing for program behaviors using heterogeneous dataflow architectures could open a new design space, ultimately reducing the importance of aggressive OOO designs and lead to greater opportunity for radical architecture innovation.

References

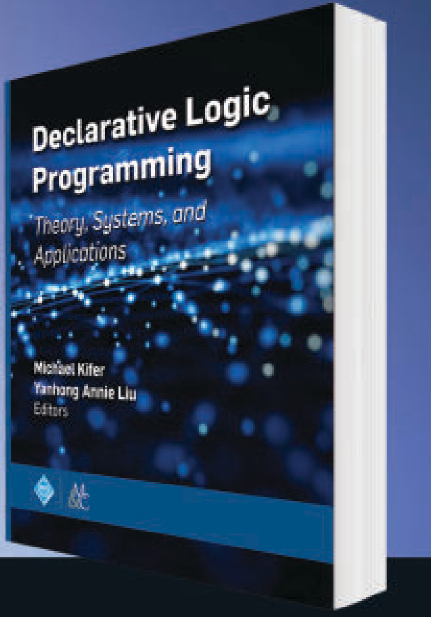
1. Arvind, K., Nikhil, R.S. Executing a program on the MIT tagged-token dataflow architecture. *IEEE Trans. Comput. 39*, 3 (1990), 300–318.
2. Budiu, M., Artigas, P.V., Goldstein S.C. Dataflow: A complement to superscalar. In *ISPASS '05 Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software* (March 20–22, 2005) IEEE Computer Society, Washington, DC, USA, 177–186.
3. Burger, D., Keckler, S.W., McKinley, K.S., Dahlin, M., John, L.K., Lin, C., Moore, C.R., Burrill, J., McDonald, R.G., Yoder, W., Team, T.T. Scaling to the end of silicon with edge architectures. *Computer 37*, 7 (July 2004), 44–55.
4. Clark, N., Kudlur, M., Park, H., Mahlke, S., Flautner, K. Application-specific processing on a general-purpose core via transparent instruction set customization. In *MICRO 37 Proceedings of the 37th Annual IEEE/ACM International Symposium on Microarchitecture* (Portland, Oregon, December 04–08, 2004), IEEE Computer Society, Washington, DC, USA, 30–40.
5. Gebhart, M., Maher, B.A., Coons, K.E., Diamond, J., Gratz, P., Marino, M., Ranganathan, N., Robatmili, B., Smith, A., Burrill, J., Keckler, S.W., Burger, D., McKinley, K.S. An evaluation of the trips computer system. In *ASPLOS XIV Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems* (Washington, DC, USA, March 07–11, 2009), ACM, New York, NY, USA, 1–12.
6. Govindaraju, V., Ho, C.-H., Nowatzki, T., Chhugani, J., Satish, N., Sankaralingam, K.,

Kim, C. DYSER: Unifying functionality and parallelism specialization for energy-efficient computing. *IEEE Micro 32*, 5 (Sept. 2012), 38–51.
7. Gupta, S., Feng, S., Ansari, A., Mahlke, S., August, D. Bundled execution of recurring traces for energy-efficient general purpose processing. In *ASPLOS XIV Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems* (Washington, DC, USA, March 07–11, 2009), ACM, New York, NY, USA, 1–12.
8. Hayenga, M., Naresh, V., Lipasti, M. Revolver: Processor architecture for power efficient loop execution. In *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)* (Orlando, FL, USA, 2014), IEEE, 591–602.
9. Kocher, P., Horn, J., Fogh, A., Genkin, D., Gruss, D., Haas, W., Hamburg, M., Lipp, M., Mangard, S., Prescher, T., Schwarz, M., Yarom, Y. Spectre attacks: Exploiting speculative execution. In *40th IEEE Symposium on Security and Privacy (S\&P'19)* (IEEE Computer Society, 2019).
10. Lee, C., Potkonjak, M., Mangione-Smith, W. MediaBench: A tool for evaluating and synthesizing multimedia and communications systems. In *MICRO 30 Proceedings of the 30th Annual ACM/IEEE International Symposium on Microarchitecture* (Research Triangle Park, North Carolina, USA, December 01–03, 1997), 330–335.
11. Li, S., Ahn, J.H., Strong, R.D., Brockman, J.B., Tullsen, D.M., Jouppi, N.P. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *MICRO 42 Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture* (New York, New York, December 12–16, 2009), 469–480.
12. Liu, Y., Furber S. A low power embedded dataflow coprocessor. In *ISVLSI '05 Proceedings of the IEEE Computer Society Annual Symposium on VLSI: New Frontiers in VLSI Design* (May 11–12, 2005), 246–247.
13. Nowatzki, T., Gangadhar, V., Ardalani, N., Sankaralingam, K. Stream-dataflow acceleration. In *ISCA '17 Proceedings of the 44th Annual International Symposium on Computer Architecture* (Toronto, ON, Canada, June 24–28, 2017), 416–429.
14. Nowatzki, T., Gangadhar, V., Sankaralingam, K., Wright, G. Pushing the limits of accelerator efficiency while retaining programmability. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, (March 12–16, 2016), 27–39.
15. Nowatzki, T., Govindaraju, V., Sankaralingam, K. A graph-based program representation for analyzing hardware specialization approaches. *Comput. Archit. Lett. 14*, 2 (July-Dec 2015), 94–98.
16. Nowatzki, T., Sankaralingam, K. Analyzing behavior specialized acceleration. In *ASPLOS '16 Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems* (Atlanta, Georgia, USA, April 02–06, 2016), ACM, New York, NY, USA, 697–711.
17. Nowatzki, T., Sartin-Tarm, M., De Carli, L., Sankaralingam, K., Estan, C., Robatmili, B. A general constraint-centric scheduling framework for spatial architectures. In *PLDI '13 Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation* (Seattle, Washington, USA, June 16–19, 2013), ACM, New York, NY, USA, 495–506.
18. Padmanabha, S., Lukefahr, A., Das, R., Mahlke, S.A. Trace based phase prediction for tightly-coupled heterogeneous cores. In *MICRO-46 Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture* (Davis, California, December 07–11, 2013), ACM, New York, NY, USA, 445–456.
19. Papadopoulos, G.M. Monsoon: An explicit token-store architecture. In *ISCA '90 Proceedings of the 17th Annual International Symposium on Computer Architecture* (Seattle, Washington, USA, May 28–31, 1990), ACM, New York, NY, USA, 82–91.
20. Swanson, S., Michelson, K., Schwerin, A., Oskin, M. WaveScalar. In *MICRO 36 Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture* (December 03–05, 2003), IEEE Computer Society, Washington, DC, USA, 291.
21. Venkatesh, G., Sampson, J., Goulding, N., Garcia, S., Bryksin, V., Lugo-Martinez, J., Swanson, S., Taylor, M.B. Conservation cores: Reducing the energy of mature computations. In *ASPLOS XV Proceedings of the Fifteenth Edition of ASPLOS on Architectural Support for Programming Languages and Operating Systems* (Pittsburgh, Pennsylvania, USA, March 13–17, 2010), ACM, New York, NY, USA, 205–218.
22. Watkins, M.A., Nowatzki, T., Carno, A. Software transparent dynamic binary translation for coarse-grain reconfigurable architectures. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)* (March 12–16, 2016), 138–150.

Tony Nowatzki ({tjn}@cs.ucla.edu), University of California, Los Angeles, Los Angeles, CA, USA.

Some work performed at University of Wisconsin - Madison.

Vinay Gangadhar and Karthikeyan Sankaralingam ({vinay, karu}@cs.wisc.edu), University of Wisconsin - Madison, Madison, WI, USA.


Declarative Logic Programming
Theory, Systems, and Applications
Michael Kifer
Yanhong Annie Liu
Editors

University of Central Missouri

Assistant Professor in Computer Science - Multiple Positions

The School of Computer Science and Mathematics at the University of Central Missouri is accepting applications for three non tenure-track positions in Computer Science at the rank of Assistant Professor. The appointment will begin August 2019. We are looking for faculty excited by the prospect of shaping our school's future and contributing to its sustained excellence.

The Position: Duties will include teaching undergraduate and graduate courses in computer science, cybersecurity and/or software engineering, and developing new courses depending upon the expertise of the applicant and school needs, program accreditation and assessment. Faculty are expected to assist with school and university committee work and service activities, and advising majors.

Required Qualifications:

▸ Ph.D. in Computer Science, Cybersecurity or Software Engineering

▸ Demonstrated ability to teach existing courses at the undergraduate and/or graduate levels

▸ Excellent verbal and written communication skills

The Application Process: To apply online, go to https://jobs.ucmo.edu. Apply to position #997335, #997819 or #998560. The following items should be attached: a letter of interest, a curriculum vitae, copies of transcripts, and a list of at least three professional references including their names, addresses, telephone numbers and email addresses. Official transcripts and three letters of recommendation will be requested for candidates invited for on-campus interview. For more information, contact:

Dr. Songlin Tian, Search Committee Chair
School of Computer Science and Mathematics
University of Central Missouri
Warrensburg, MO 64093
(660) 543-4930
tian@ucmo.edu

Initial screening of applications begins May 1, 2019, and continues until position is filled. AA/EEO/ADA. Women and minorities are encouraged to apply.

UCM is located in Warrensburg, MO, which is 35 miles southeast of the Kansas City metropolitan area. It is a public comprehensive university with about 13,000 students. The School of Computer Science and Mathematics offers undergraduate and graduate programs in Computer Science, Cybersecurity and Software Engineering with over 700 students. The undergraduate Computer Science and Cybersecurity programs are accredited by the Computing Accreditation Commission of ABET.

U.S. NAVAL RESEARCH LABORATORY

Superintendent
Information Technology Division
www.nrl.navy.mil

Senior Executive Service Career Opportunity – Tier 2

ES-0180/0854/0855/1310/1520/1550: $126,148 - $189,600 per annum* (2019 salary)
*Actual salary may vary depending on the scope and complexity of the position and the qualifications and current compensation of the selectee.

Become a member of an elite research and development community involved in basic and applied scientific research and advanced technological development for tomorrow's Navy and for the Nation.

The Superintendent of the Information Technology Division is responsible for the conception, planning and formulation of the scientific program of the Division in pursuance of the military defense of the United States. He/she provides the Technical and administrative leadership required to insure that significant and productive accomplishments flow from that program.

As the Superintendent, you will:

- Establish priorities by considering the relative importance of the work to other work carried on by the Division, its utility and adaptability to the national defense, its usefulness in other fields of research and development, and its potential value in relation to the increase of scientific knowledge.
- Assigns problems to the operating Branches, approves their general plans, stimulates interest and activity on the part of Division personnel, reviews progress and completed work for soundness of approach, validity of conclusions and advisability of recommendations.
- Coordinates the research and development of the Division with that performed in other Divisions at the NRL, in other US and allied Government laboratories, universities and industrial laboratories.
- Is a recognized authority in the field of computer science, communications, cyber security, artificial intelligence and allied disciplines.
- Responsible for planning and directing the effective administration of the Division which includes administering budget matters that fall within the province of the Divisions operations.

Applicants should be recognized as national/international authorities and should have planned and executed difficult programs of national significance or specialized programs that show outstanding attainment in their field of research.

For more information and specific instructions on how to apply, visit www.usajobs.gov, log in and enter the following announcement number: DE-10493155-19-JS. The announcement closes 28 June 2019. Contact Lesley Renfro at Lesley.renfro@nrl.navy.mil for more information. E-mailed resumes cannot be accepted.

NRL is an Equal Opportunity Employer

NRL – 4555 Overlook Ave SW, Washington DC 20375

acm

ADVERTISING IN CAREER OPPORTUNITIES

How to Submit a Classified Line Ad: Send an e-mail to *acmmediasales@acm.org*. Please include text, and indicate the issue/or issues where the ad will appear, and a contact name and number.

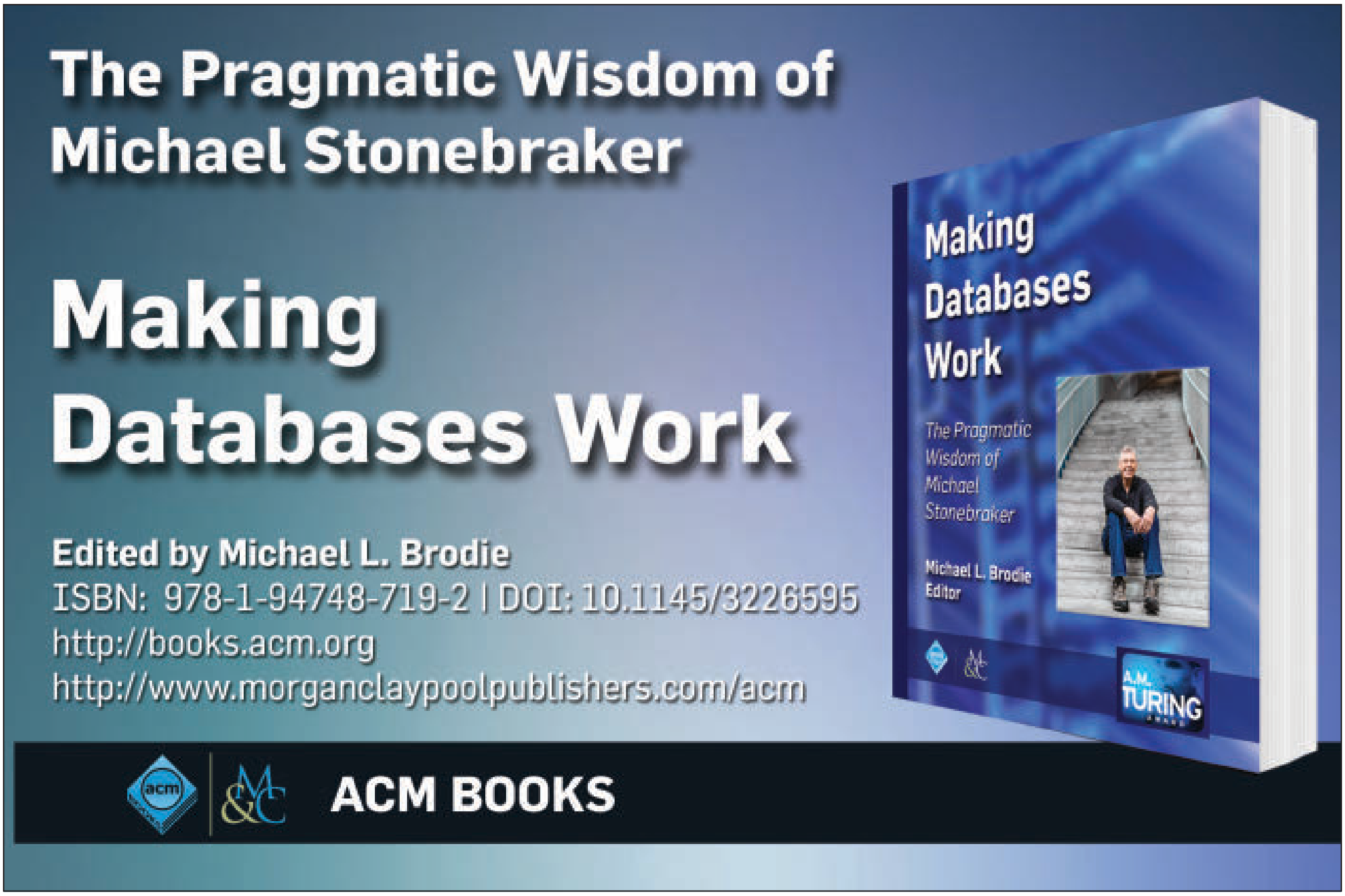
Estimates: An insertion order will then be e-mailed back to you. The ad will by typeset according to CACM guidelines. NO PROOFS can be sent. Classified line ads are NOT commissionable.

Deadlines: 20th of the month/2 months prior to issue date. For latest deadline info, please contact:
acmmediasales@acm.org

Career Opportunities Online: Classified and recruitment display ads receive a free duplicate listing on our website at:
http://jobs.acm.org

Ads are listed for a period of 30 days.

For More Information Contact:
ACM Media Sales
at 212-626-0686 or
acmmediasales@acm.org



Making Databases Work
The Pragmatic Wisdom of Michael Stonebraker
Michael L. Brodie
Editor
A.M. TURING AWARD

[CONTINUED FROM P. 96] keep the connection alive with people who are trying to understand how the brain works.

HINTON: That said, neuroscientists are now taking it seriously. For many years, neuroscientists said, "artificial neural networks are so unlike the real brain, and they're not going to tell us anything about how the brain works." Now, neuroscientists are taking seriously the possibility that something like backpropagation is going on in the brain, and that's a very exciting area.

LECUN: Almost all the studies now of human and animal vision use convolutional nets as the standard conceptual model. That wasn't the case until relatively recently.

HINTON: I think it's also going to have a huge impact, slowly, on the social sciences, because it's going to change our view of what people are. We used to think of people as rational beings, and what was special about people was that they used reasoning to derive conclusions. Now we understand much better that people are basically massive analogy-making machines. They develop these representations quite slowly, and then the representations they develop determine the kinds of analogies they can make. Of course, we can do reasoning, and we wouldn't have mathematics without it, but it's not the fundamental way we think.

For pioneering researchers, you seem unusually unwilling to rest on your laurels.

HINTON: I think there's something special about people who invented techniques that are now standard. There was nothing God-given about them, and there could well be other techniques that are better. Whereas people who come to a field when there's already a standard way of doing things don't understand quite how arbitrary that standard way is.

BENGIO: Students sometimes talk about neural nets as if they were describing the Bible.

LECUN: It creates a generation of dogmatism. Nevertheless, it's very likely that some of the most innovative ideas will come from people much younger than us.

The progress in the field has been amazing. What would you have been surprised to learn was possible 20 or 30 years ago?

LECUN: There's so much I've been surprised by. I was surprised by how late the deep learning revolution was, but also by how fast it developed once it started. I would have expected things to happen more progressively, but people abandoned the whole idea of neural nets between the mid-1990s and mid-2000s. We had evidence that they were working before, but then, once the demonstrations became incontrovertible, the revolution happened really fast, first in speech recognition, then in image recognition, and now in natural language understanding.

HINTON: I would have been amazed, 20 years ago, if someone had said that you could take a sentence in one language, carve it up into little word fragments, feed it into a neural net that starts with

PHOTO BY ALEXANDER BERG

random connections, and train the neural net to produce a translation of the sentence into another language with no knowledge at all of syntax or semantics—just no linguistic knowledge whatsoever—and it would translate better than anything else. It's not perfect, it's not as good as a bilingual speaker, but it's getting close.

LECUN: It's also amazing how quickly these techniques became so useful for so many industries. If you take deep learning out of Google or Facebook today, both companies crumble; they are completely built around it. One thing that surprised me when I joined Facebook is that there was a small group using convolutional nets for face recognition. My first instinct about convolutional nets was to think they would be useful for, maybe, category-level recognition: car, dog, cat, airplane, table, not fine-grained things like faces. But it turned out to work very well, and it's completely standard now. Another thing that surprised me came out of Yoshua's lab on generative adversarial networks—that you can basically use neural nets as generative models to produce images and sound.

BENGIO: When I was doing my Ph.D., I was struggling to expand the idea that neural nets could do more than just pattern recognition—taking a fixed-size vector as input and producing categories. But it's only recently with our translation work that we escaped this template. As Yann said, the ability to generate new things has really been revolutionary. So has the ability to manipulate any kind of data structure, not just pixels and vectors. Traditionally, neural nets were limited to tasks that humans can do very quickly and unconsciously, like recognizing objects and images. Modern neural nets are different in nature from what we were thinking about in the 1980s, and they can do things that are much closer to what we do when we reason, what we do when we program computers.

In spite of all the progress, Yoshua, you've talked about the urgency of making this technology more accessible to the developing world.

BENGIO: I think it's very important. I used to not think much about politics, but machine learning and AI have come out of universities, and I think we have a responsibility to think about that and to participate in social and political discussions about how they should be used. One issue, among many, is where is the know-how and wealth and technology are going to be concentrated. Are they going to be concentrated in the hands of a few countries, a few companies, and a small class of people, or can we find ways to make them more accessible, especially in countries where they could make a bigger difference for more people?

HINTON: Google has open-sourced its main software for developing neural nets, which is called TensorFlow, and you can also use the special Google hardware for neural nets on the cloud. So Google is trying to make this technology accessible to as wide a set of people as possible.

LECUN: I think that's a very important point. The deep learning community has been very good at promoting the idea of open research, not just within academia, where conferences distribute papers, reviews, and commentaries in the open, but also in the corporate world, where companies like Google and FB are open-sourcing the vast majority of the software that they write and providing the tools for other people to build on top of it. So anyone can reproduce anyone else's research, sometimes within days. No top research group is ahead of any other by more than a couple of months on any particular topic. The important question is how fast the field as a whole is progressing. Because the things we really want to build—virtual assistants that can answer any question we ask them and can help us in our daily lives—we just don't just lack the technology, we lack the basic scientific principles for it. The faster we can foster the entire research community to work on this, the better it is for all of us. C

Leah Hoffmann is a technology writer based in Piermont, NY, USA.

Watch the recipients discuss this work in the exclusive *Communications* video. https://cacm.acm.org/videos/2018-acm-turing-award

DOI:10.1145/3324011

Leah Hoffmann

Q&A
Reaching New Heights with Artificial Neural Networks

ACM A.M. Turing Award recipients Yoshua Bengio, Geoffrey Hinton, and Yann LeCun on the promise of neural networks, the need for new paradigms, and the concept of making technology accessible to all.

ONCE TREATED BY the field with skepticism (if not outright derision), the artificial neural networks that 2018 ACM A.M. Turing Award recipients Geoffrey Hinton, Yann LeCun, and Yoshua Bengio spent their careers developing are today an integral component of everything from search to content filtering. So what of the now-red-hot field of deep learning and artificial intelligence (AI)? Here, the three researchers share what they find exciting, and which challenges remain.

There's so much more noise now about artificial intelligence than there was when you began your careers—some of it well-informed, some not. What do you wish people would stop asking you?

GEOFFREY HINTON: "Is this just a bubble?" In the old days, people in AI made grand claims, and they sometimes turned out to be just a bubble. But neural nets go way beyond promises. The technology actually works. Furthermore, it scales. It automatically gets better when you give it more data and a faster computer, without anybody having to write more lines of code.

YANN LECUN: That's true. The basic idea of deep learning is not going away, but it's still frustrating when people ask if all we need to do to make machines more intelligent is simply scale our current methods. We need new paradigms.

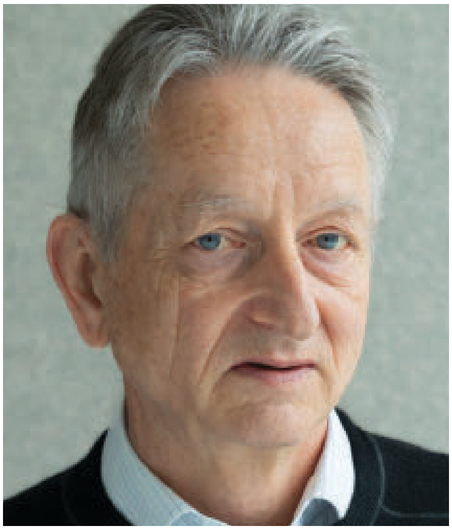
YOSHUA BENGIO: The current techniques have many years of industrial and scientific application ahead of them. That said, the three of us are researchers, and we are always impatient for more, because we are far from human-level AI, and the dream of understanding the principles of intelligence, natural or artificial.

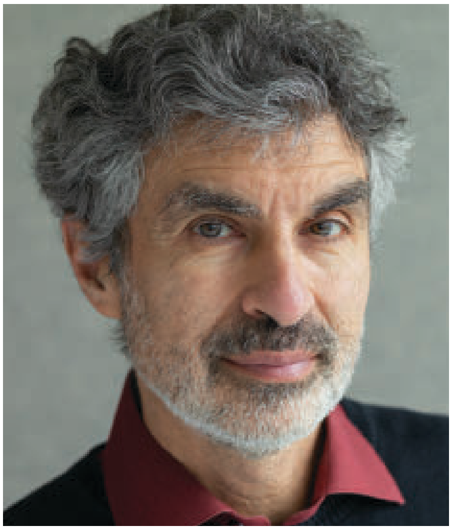
What isn't discussed enough?

HINTON: What does this tell us about how the brain works? People ask that, but not enough people are asking that.

BENGIO: It's true. Unfortunately, although deep learning takes inspiration from the brain and from cognition, many engineers involved with it these days don't care about those topics. It makes sense, because if you're applying things in industry, it doesn't matter. But in terms of research, I think it's a big loss if we don't

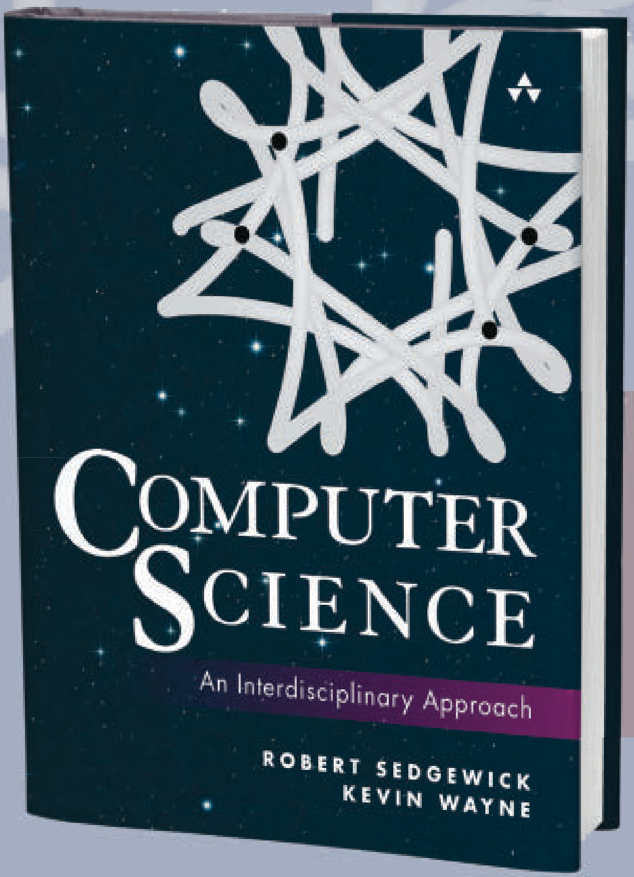
[CONTINUED ON P. 94]


Geoffrey Hinton


Yoshua Bengio

Yann LeCun

PHOTOS BY ALEXANDER BERG


COMPUTER SCIENCE
An Interdisciplinary Approach
ROBERT SEDGEWICK
KEVIN WAYNE

Algorithms
FOURTH EDITION
ROBERT SEDGEWICK | KEVIN WAYNE