Roundtable

Autonomous Systems and the Challenges in Verification, Validation, and Test

by David Yeh

There is no question that machine learning is a hot topic with potential to change the way humans interact with the world around them. Digital assistants have become common, and applications tout the term as if it conveys magical properties. Underneath all the hype, though, lies a serious problem. With unsupervised learning just around the corner and the next leap forward needed to enable autonomous systems, what will the design community do to handle the verification, validation, and test of such systems? This roundtable assembled a group from academia, industry, and government to discuss key issues related to this topic and covers aspects of training, data, interpretability, and electronic design automation.

IEEE Design&Test thanks the participants at the 2017 Design Automation Conference roundtable: moderator Sankar Basu (National Science Foundation), Sanjit A. Seshia (University of California, Berkeley), Jenna Wiens (University of Michigan), Li-C. Wang (University of California, Santa Barbara), and Ruchir Puri (IBM). D&T gratefully acknowledges the help of David Yeh (SRC), our roundtables editor, who also participated in the event.

Digital Object Identifier 10.1109/MDAT.2018.2816940
Date of current version: 21 May 2018.

Sankar Basu: Autonomous machine-learning systems are increasingly pervasive in our world: we have, or will have in the foreseeable future—self-driving cars, drones, unmanned submarines, and surgical robots—to mention only a few. As we've learned at this conference, these new systems depend on unsupervised machine learning. I view autonomous systems as an attempt to bring test, verification, and machine learning together within the larger context of cognitive computing introduced by Puri. Then, if we consider Wang's test approach and the issues raised by Seshia and Wiens in earlier presentations, we can extract considerable information from those and related research.

Ruchir Puri: Each of us has outlined some common issues related to machine learning. First, we have "brute forced" the learning/training problem today to a large extent, which has certainly resulted in tremendous progress over the past 5 years, yet in many ways we don't have enough data in, for example, medicine and healthcare as Wiens pointed out. You know, 100,000 is a large context in medicine. For various reasons, we don't have that data, and dealing with smaller data is a problem that we need to deal with because cognition is learning with smaller data more than learning with larger data.

Copublished by the IEEE CEDA, IEEE CASS, IEEE SSCS, and TTTC

A second issue involves reasoning and what I call *why*. Techniques are very much "black-boxed," and most people, specifically in healthcare and enterprise situations, want to know *why* if something were to go wrong.

A third issue is robustness. Techniques need to be robust. I think adversarial training techniques in deep learning certainly have made excellent progress. Still, robustness of techniques remains a critical challenge.

Finally, the fourth issue is computing—graphics processing units certainly have made things easier; however, they were not designed for this type of computation. We see a lot of activity in this area now, but we still lack ideas on nontraditional architectures. IBM's TrueNorth was an example, but it only does inferencing; it doesn't do learning and training.

Basu: Are you saying that there is a lot of data available in healthcare, but not in other enterprise applications?

Puri: No. I'm saying even in healthcare the amount of labeled data, where we know a lot about the structure and format of the information, is not that large, actually. In consumer domains, labeled data is easy to come by, because you can have literally billions of people give you feedback via Google, for instance. It usually doesn't happen in enterprise situations. Enterprises have a lot of data, but they don't have the explicitly labeled data, the kind you need for generating artificial intelligence (AI) algorithms with supervision learning. AI is nothing without data, which is the critical bottleneck.

Jenna Wiens: To add to that, it is not just about getting but about sharing data. Reproducibility is a key issue in machine learning for the healthcare community. While I can access data, I cannot necessarily share those data. Benchmark data sets that exist in the computer vision and the natural language processing (NLP) communities have been critical to the advancement of those fields. Such data sets don't yet exist to the same extent in healthcare.

Basu: Even many years ago when I worked on speech and imaging, it was said that "more data is gold." Nothing is like more data—still true today. On the other hand, deep learning is probably successful due to a huge amount of data and computer power available. Yet have we made any advances in basic understanding as to why deep learning works and what we are doing rather than just throwing big machines and more data at it? Will customers trust what's essentially a black box?

Sanjit A. Seshia: I think there are two potential approaches to that question. On the one hand, there are some very rigorous, mathematical—although mostly "paper and pencil"—analyses regarding machine learning. On the other hand, from a design automation perspective, we can apply automated techniques for analyzing machine learning systems and synthesizing models of these very complex systems that are approximate by nature, but which can explain what they're trying to do. For example, in inductive program synthesis we have a template structure for the program and can synthesize it from examples. I'm using the word program very broadly here; think of the deep learning network as a function from the input layer to the output layer. Now, instead of thinking of it as a complete function given by the network, we can hypothesize it as a particular kind of partially specified function and write an incomplete program that has some high levels or structure of rules. Then, we can synthesize the best approximation among those programs to the actual artifact, and the program we get at the end is interpretable. However, it is an approximation, so we have to find a balance. In general, this area of inductive synthesis is something that we in the design automation (especially the verification and synthesis) community can offer on this particular topic.

Basu: So, this is a research agenda on explainable AI for the future.

Seshia: Yes, it's a research agenda, and "explainable AI" is part of it. On a related note, an MIT project recently showed a result on learning from a small number of examples. They applied an inductive synthesis approach to learn a model from only a few examples, just as a human would, but which could achieve accuracy similar to those conditions that machine learning techniques based on a lot of data could achieve. More generally, we could have some expert-provided structure, in the form of a program template, which is essentially an encoded domain structure that could be used with a few well-chosen examples. Additionally, in this setting, the role of an oracle that chooses examples becomes important. I call this oracle-guided inductive synthesis.

Basu: In the image recognition community, there's the problem of one-shot learning, so what you're describing sounds like learning with very little data. In fact, your colleagues at University of California, Berkeley and Stanford have a project, which is exploring that learning paradigm in both software and hardware architectures based on these neuroscience-inspired techniques—they called it hyperdimensional learning, geared toward this one-shot or little-data learning approach.

Puri: Yes! One-shot, and multishot, learning as well. Transfer learning is another way to learn in one domain, and we can transfer that learning without requiring a huge amount of data in that domain, as long as the domains are adjacent. We're using both those techniques, beginning to move from research into production. I still feel over-reliance on very large data sets, specifically spurred by the catalyst of ImageNet, has hampered us in some ways, because we don't see acquiring that much training data from enterprises. It is very hard to come by.

Basu: Puri, as someone more involved with practice, would you say there will be a time when more training data will no longer be available?

Puri: That moment is here now. Most of our setup time involves acquiring the training data. It involves acquiring the training data, removing noise from data, refining data, and so on. If I look at end-to-end systems and bringing AI to enterprises, data is the bottleneck. It is always been, as you rightly pointed out. But in some domains, we've addressed it by sheer brute force to put the ImageNet label sets together. Some companies—Amazon Mechanical Turk, Mighty AI, CrowdFlower, and others—just outsource this. They've built very good tooling, but they're not experts in industry-vertical domains. You cannot outsource expert medical opinions to just anyone out there, which is why I stress that in enterprises, data for training is a bottleneck.

Wiens: Even when you collect the data, you don't just do it once, because the environments from which you collect the data are not fixed. Things change: in a hospital or healthcare setting, patient populations, treatments, clinical protocols, and even hospital layouts can all change, which typically requires us to collect additional data.

Li-C. Wang: The learning model improves when the data grows very, very large—in theory, if your learning algorithm follows Occam's razor and the accuracy continues to improve, then you can prove you're actually learning something. That's the theoretical side. Researchers have observed improved accuracy in the past 5 years, from an experiment with a very large, deep neural network, on the basis of the theoretical prediction. If we keep going that way—improving accuracy by processing more data—then eventually we'll reach 99.9% of accuracy. There's a theoretical reason for that, but for limited data it can be debatable, theoretically, as to what machine learning means.

Seshia: If you look broadly at the machine-learning literature, there are approaches like Bayesian inference and deep learning that dominate conferences such as International Conference on Machine Learning today, but there are also the computational learning theory conferences and the literature from 20 to 30 years ago on topics such as query-based learning. We can think of these learning algorithms as learning from small data. It includes active learning but goes well beyond it to oracle-guided learning and machine teaching.

Wiens: Well, active learning and reinforcement learning.

Seshia: We could include reinforcement learning, yes. But the setting I have in mind is the "teaching" case where the teacher knows what the concept is, and they can only communicate it to students through examples, in response to queries. Then, what's the smallest number of examples we need to teach a concept? If we need a huge number of examples, we have no hope of doing any better in active learning. There have been studies on this topic with some interesting results, but much more needs to be done.

Wang: You mentioned computational learning theory. That's one good example, but in computational learning theory what it proves is that unless it is for a very simple function we can learn, most of the function is not efficiently learnable—most of the learning problems are NP- or crypto-hard.

Seshia: I know, but we come from the DAC community where we think SAT (Boolean satisfiability) is easy! If we have oracles that can solve SAT problems, the question is not whether we need an exponential number of examples, but how many oracle queries do we need? If the number of queries to the oracle is phenomenally small, it is a tractable problem.

Wang: Exactly. We don't understand the theoretical side to answer that small-data learning problem well enough. A lot of techniques we apply are like reinforcement or transfer learning. They may show good results in practice on some benchmarks, but they are more *ad hoc*. There's no assurance where that's going to leave us if we keep pushing to other benchmarks. Until we answer the theoretical question of how to learn with small data, we don't know how a particular learning algorithm designed for big data will perform on a given small data set.

Seshia: I agree: we need to do more work on this. Recently, we've been trying to gather people from both the machine learning and the formal methods

communities so that we can grow the community of people working on exactly this topic.

Basu: Is this issue somehow related to the "master class learning" technique—using hidden or privileged information—as advanced by Vladimir Vapnik in recent years?

Wang: Yes. The neural network and natural languages efforts have been so successful that the potential applications are many. For example, the next thing we're going to see is the personal assistant. Right now I can talk to my cellphone; have it remind me of a meeting tomorrow afternoon, say, or I can ask a question about something. The technology, however, is really speech processing. From the revenue perspective, those two things we talked about, computer vision and NLP, in the next 5 years will still be driving the market.

Then, there are other things such as business intelligence. I work in manufacturing intelligence, and as in other areas, we don't have big data. Also, a lot of people don't want to share data because there are a number of manufacturing secrets, and I don't think that would change in the near term unless there's an incentive to save, say, $500 million. For machine learning, the low-hanging fruit in the next few years will still be computer vision and NLP.

Seshia: There are two broad classes of machine learning. One is when it's used in an engineered system, like a self-driving vehicle, where machine learning is one component of that system. The second is in something like personalized medicine where the learning system is based on a limited amount of data about a natural, nonengineered system and that can be used as a tool for doctors, nurses, or even patients to make decisions. It is useful to distinguish between the two classes because in the case of engineered systems, simulation or some sort of design automation tools allow us to collect data with reasonable models in addition to collecting data in the real world. Then, we can systematically explore system behaviors and gain more confidence before going out on the road.

Basu: Perhaps, because there is a physical model underlying all of that.

Seshia: Exactly. These sorts of physics-based simulation techniques and associated rendering engines have gotten so much better for robotics and cyber-physical systems (CPSs). On the other hand, for the field of medicine, we don't have good models of ground truth, outside of what the doctors, biologists, and medical staff have in mind, so it is harder to do something based on simulation. Maybe in restricted settings, work in systems biology has resulted in models where we could acquire the necessary data, but it seems harder. With engineered systems, we can extensively use modeling, simulation, and other verification techniques—we can build better verification and testing tools that help us solve the problem of finding the right or small data that we need to collect.

Basu: In a sense, what you're saying is that for these engineered systems, because there is an underlying physical model, we can (or try to) solve the small-data problem to some extent, while it is harder in the other case.

Seshia: Right. Those systems also let us make progress on the verification and testing problem in simulation first to catch a lot of the "shallow" bugs. Then, we can use the analysis for targeted field and road testing to catch the harder bugs.

Basu: Maybe there's something fundamental I don't understand about this bug catching in the machine-learning scenario. Oversimplifying things to some extent, I can think of two kinds of algorithms. One kind is the deterministic, support vector machine type of algorithm, which is a classifier. We could have overlapping classes, where the decision boundary is not always clean, and where test data classification does not fall neatly into class A or class B. The problem is that we have an opportunity to break the system at the very outset, because there will be examples that will be wrong, but we try to reduce the expected risk.

With the other kind of problem—speech recognition, say—we're using basically a communication theory model. With that, we try to reduce the probability of error, but there'll be cases where we will go wrong. How can we ultimately verify things in the same way that the model-checking people would?

Seshia: We should start with a system-level specification. Using traditional, purely data-driven approaches, the machine learning classifier is going to be wrong in some cases, because data is incomplete and ground truth is hard to define—maybe undefinable in many cases. But we don't care about misclassification if, at the system level, the desired property is satisfied. In fact, in our work verifying autonomous driving systems, we found misclassifications that don't matter, because the

controller had logic in it to correct for it. A lot of the autonomous driving controllers don't use just one sensor input; they use multiple sensors. They fuse the data. So even if there was a misclassification on one of those things—say, the visual sensor—the controller may still be robust to make a needed correction. Or your car may have vehicle-to-vehicle or vehicle-to-infrastructure communication that tells it an obstacle is coming up even though its sensor has malfunctioned. I think that's the key. We don't care about misclassifications as long as the system-level specification is satisfied.

Basu: Is this redundancy of sorts, which mitigates the problem of misclassification?

Wang: Even when we look at one of today's designs, including the whole system, we aren't going to see a bug-free design because it is too complex—nobody can guarantee that it is going to be 100% bug-free. So how do system designers mitigate the potential bugs? When they discover something is questionable, they just hide it at the firmware level, for example. They can just avoid that usage even though the hardware allows it. From the user perspective, we don't see any hardware bug because it is been mitigated and hidden from the software level. I may have a perception that a component is not 100% correct, but I can say, "I have an intelligent controller which doesn't just take the result from a component and use it; it will evaluate situations and then make a safety evaluation." If something isn't safe, the controller's going to reject it. What we really care about is if the total system meets a spec. Internally, it may not be 100% correct.

Basu: Going back to my IBM and speech recognition days, the speech was recognized then not only as acoustic speech but also from lip reading, which is image processing. If one system gave us a wrong answer, the other could mitigate the error.

Wiens: You mentioned that internally the system is not 100% correct. Well, in healthcare it is not 100% correct externally either. That's where physicians come in. Having that expert in the loop is still crucial.

Basu: And the input data can be wrong as well.

Wiens: Oh, absolutely. Or the features used as input to the model are inaccurate. In terms of predictive performance, is it possible to achieve an area under the receiver operating characteristics curve of one in healthcare? In many applications, no. It is just not believable.

Basu: So what verification means in this context also needs to be clarified.

Seshia: In healthcare, it is very challenging. The verification problem that I've thought about in healthcare is for robotic surgery. It is closer to an engineered system, but it is different from self-driving vehicles or drones. There's a human in the loop with a lot more control over many aspects, but they have to train in a simulator first. Often, they don't get enough feedback from the simulator, so the doctors that use this system aren't completely comfortable at first, and they grow their trust in the surgical robot over time. In a self-driving vehicle, there may be similar situations with a safety driver, but eventually, when it is in autonomous mode, the car is driving itself.

With a drone, there's even more autonomy since the person piloting the drone once it goes into autonomous mode is just giving waypoints and letting it go. What's interesting is that a lot of the systems that use learning have humans in the loop, and I think this is going to be the case for some time.

When we talk about verification of these systems, we cannot ignore the human in the loop because, as we know from avionics, a lot of the problems arise from the so-called automation surprises. The pilot thought that the mode was one thing when it was actually something else; they make the wrong decision that leads to an accident or near-miss. There needs to be more work on modeling the human from a verification and modeling perspective and then using those models to design the controller for these systems as well as the human interface in a way that's natural.

Puri: Interestingly, AI will be conservative in many use cases. For example, autonomous vehicles in cities like Boston or New York would have a hard time. The way the cars are designed, they wouldn't move in those cities because they are expecting a person to behave a certain way. Pedestrians won't stop many times even if the crossing light is red, so in cities like Boston or New York that are so densely populated, pedestrians' interaction time is much shorter than, say, in Phoenix. As humans, we have become much better at adapting—we also take risks and make calculated decisions—but the self-driving vehicles don't.

Wiens: Models also can affect one's behavior. Once we deploy a model at a hospital, and physicians start acting on that model, it will shift the data.

And how to tease apart what has shifted—because of human action, because of the model feedback—versus a shift due to changes in clinical protocol is an interesting technical challenge, one that's necessary to solve if we want these models to safely adapt over time. But even simply considering how people will interact with systems and how will it change their behavior is interesting.

Puri: AI-driven systems, especially self-driving cars, are risk averse and in real tight human environments wouldn't work, because, yes, we do take risks. We do make calculated decisions.

Seshia: This brings up an interesting point on specifications and assertions. If we very conservatively write rules of the road for autonomous vehicles that they shouldn't break, then they'll freeze, right? But humans know when to break the rules and when to stick to them. We can get into a situation with an unprotected left turn and we just go with a riskier turn because it is unsafe to stay where we are. But an autonomous vehicle may not do that. What that means to me is that we need to move from a Boolean notion to a quantitative notion of correctness. It could be probability or just some kind of score, and we use that in determining if an assertion is satisfied before making the decision. In the formal methods community, there's a trend toward assertion logics that have this sort of quantitative semantics dealing with the degree of satisfaction, not just whether it's true or false. That's a trend that could, down the road, yield some interesting theory for us to build on.

Basu: Some of these systems, including the autonomous vehicle, could be analog, digital, or hybrid. Even neural networks can be digital or analog or a mixed type, so does that pose some additional challenges for new research?

Seshia: It does. Fortunately, the hybrid systems community has been working on these problems for 20+ years, and at this point, we have tools that can operate on industrial designs. Automotive companies are using some of these tools, which involve simulation-driven verification, on production designs. We might have some humongous Simulink or MATLAB model, with embedded C or C++ code and the model of the physical system—it is not a hybrid automaton style model. But we can simulate it and write temporal logic properties, and verify those in simulation. It is very similar to the workhorse of digital design; it is assertion-based verification based on simulation and already in industrial use. What hasn't gotten there, of course, is model checking, and the analog of SAT solving, and so on. There's lots of work to be done to make those tools apply to industrial CPSs.

Basu: Is there no model checking for the hybrid systems?

Seshia: There is, but a lot more progress is required. First of all, most problems are undecidable, so we can only give one-sided answers. Second, we can apply abstraction-based techniques, but they don't scale as well as what we have for digital systems and apply only to certain subclasses of hybrid systems.

Wang: It's interesting you mentioned the analog solution for neuromorphic systems. Actually, analog systems have more flexibility to optimize, so sometimes they can achieve lower power consumption and become more power efficient. But they raise difficulty in verification and test because involves analog. Everything is harder in terms of verification and test. The tradeoff is that if we do more optimization we can get better performance, but we have a harder verification and test problem.

Also, in the machine learning world there is this tradeoff related to the complexity of our model and the model interpretability. As the model complexity increases, it becomes a more powerful model but also less interpretable, and thus possibly less reliable. This is a tradeoff, so perhaps the verification need will eventually become a constraint for how complex a model can be. In a sense, we have to understand the ability of verification and what design we can verify. Verification provides constraints to specify what types of neural networks we can verify. Otherwise, we can do more optimization in an experimental setting, but we'll never be able to commercialize that idea, because we're not going to pass the safety requirement.

Seshia: At some point, machine learning systems have to be designed for verification. Right now, I don't have enough understanding of what works well (and what doesn't) in the machine-learning context to be able to suggest how to design for such verification.

Puri: We have to turn around our thinking to bring in risk as part of verification, which is a very upside-down notion: risk is part of verification. These things will take risks, otherwise they will not work: we will be deadlocked. Risk is part of verification and that's what we humans do—we take risks. Sometimes things go wrong, but in the

bigger scheme of things, they usually work out. It's a very interesting thought—risk being a part of verification.

Wiens: It's a cost-benefit analysis.

Seshia: Exactly. What makes it also interesting is that for a lot of the verification work in the circuits world we talk about the problem not as risk mitigation, but as to whether it's correct or not. If verification deals with correctness, then the underlying problem becomes satisfiability and so on. But if verification becomes risk mitigation, the underlying problem becomes multi-objective optimization. We don't really have all the tools to solve these optimization problems now, so there is an opportunity for further work. There's also opportunity in terms of thinking about the specifications: when can I turn something into a cost function and when should it really be more binary? The choice of which properties we can tradeoff and which we cannot is difficult.

Wiens: It's very domain dependent.

Wang: It's interesting you mentioned risk, because from the test perspectives, that's already a factor—for example, test for the automotive market requires something like 0.1 defective parts per million. That's a risk, but how you actually calculate (or guarantee) that is the problem. I think for this kind of verification, once we add risk, the issue becomes more like a test kind of concept, and why the boundary between these two may not be so rigid.

Seshia: Consider the IBM Watson system. There are so many different domains and for certain kinds of questions in those domains, you might not need very high accuracy, but for others you certainly do. What is the current thinking on how to approach that situation, and how do you quantify it?

Puri: We approach it by brute force. Models are built for individual domains. We have multiple layers, so we have a public model, a domain model, and a private model. Verizon's model is not shared with AT&T, and so on. This differs from Google, because if I type in a Google query, and you make a query a second later than I did, you likely benefited from my query. In our case, however, we're dealing with all enterprise situations.

Basu: On another topic, which Puri alluded to earlier today: One of the reasons that someone would choose a neural-network type of architecture could be that a lot of data is available. These architectures are originally inspired by the brain, which as we know operates at very low power. Power is a motivation for looking into the neural-network type of architecture. On the other hand, with neural networks we don't typically worry about verification or all these other issues. But if we add verification to the mix, we may not save power anymore. Will that defeat the whole purpose?

Puri: If verification is required, then it might be that the overhead is large enough that you may not really save much power. Assume we are going to build models that have risk-based verification built in. Then assume there is a bigger function whereby somebody is measuring system effectiveness, and yes, they are either at par with how the humans behave, or better. What does it mean to be at par with humans or better? The model itself can be implemented in hardware, whether directly mapped onto a special-purpose architecture or a more general-purpose architecture. That model is for inferencing, and inferencing in itself can map into highly power-efficient architectures. The problem comes with architectures for training, which are much more power hungry, and if we add verification, the problem becomes even more complicated. First, we are nowhere near to addressing the training problem with respect to current domains, leaving aside risk verification.

Basu: But that training can be offline, right?

Puri: It can be offline, but training is also continuous as well. A significant part of it is online training where we are always learning.

Wang: That's an interesting—but really open—question, because if that's true, we're really showing there's no free lunch between efficiency and verification.

Seshia: My view is slightly contrarian because I think verification could help the training phase be less expensive by providing the right data. I don't have a lot of results to back this up yet, but we do have some work in progress, trying to use what I had mentioned in my talk to generate interesting images, which you then use to augment your data set and retrain, rather than going out and collecting another 1000 or 10,000 images. You can gain as much accuracy as you did when learning from all of the additional collected images.

Wiens: But that's still domain dependent, so maybe in that particular domain you'll have the rotational invariance, but what are the invariances in healthcare that we can exploit?

Seshia: The tools should help us find them.

Wang: Seshia's technique is more abstract in the sense it can be applied to many domains, and I think

what he's saying is that he might be able to explore the space in a formal way and identify the key sample, then just add onto the training set.

Seshia: Right. This is like using the verifier as an oracle or a teacher that gives you just the right examples you want to learn from to avoid unsafe situations.

Wang: The fundamental question will be whether that computation will outperform the computation where you actually train in the conventional way.

Seshia: If you have a safety-critical system, you've got to run the verification anyway.

Wang: Yes, because the tail sample is very rare, and we won't be able to identify them without these types of techniques.

Puri: So does this link it back to an earlier point, using some of these techniques that Seshia is talking about, linking it to smaller data, which are more power-efficient because they don't need to process 14 million images? That combined system will be more power efficient, which is the point you are making. And, again, considering Wiens's point, I agree that would be very domain dependent given just the variety of users I see in various industries. It may be easier to say that, yes, these techniques are domain specific, and more abstract as well, but it's hard for me to see right now, because these examples that you generate—what's the right example to use when teaching, as Wiens pointed out, may be very domain dependent? For images it may be relatively easier; in other situations it's very unclear.

Seshia: Exactly. The verification technique, the approach, the algorithms are general. But the data must be generated in a way that is not just domain dependent but also system dependent. We could be verifying a particular system and generate examples relevant for that specific design, but if we have to do verification ultimately anyway, why not leverage that to reduce the training burden as well? If verification isn't needed ultimately, maybe other techniques are better.

Basu: This is intriguing, but I am not sure to what extent we know how to do this.

Seshia: It's a research problem, a promising research direction.

David Yeh: It's like asking what could go wrong, for instance, in processing an image. If I can see an image with a triangle, and there's a bird in it, maybe that's a birdhouse. So you have to ask yourself what could go wrong.

Seshia: Would the self-driving car collide because of that? Maybe not because it sees the birdhouse as an obstacle. The sign on the pole is also an "obstacle." Does it have to interpret that sign correctly to stay safe? Maybe not. That's the key—is adding that extra image to the data set adding value in terms of making your system safer? If not, then we want to search for those images that do add value.

Yeh: And in the healthcare space we've got this model where we're asking, "Well, if I'm inaccurate in one particular vector or variable, what does that do to my outcome and my risk?" That's very hypothetical. If my blood pressure is 10 points higher, what does that do to my risk?

Basu: Speaking of healthcare, I have a question for Wiens: Interpretability—unless, of course, I misunderstood—means that if you have an outcome that is not interpretable, you're going to try to force your system to make it interpretable, because otherwise medical practitioners do not accept that outcome. Perhaps this is where the medical sciences and the healthcare industry differ from physical sciences: in physics, if computation produces something that the physicists cannot explain, they go and look for new physics. Even in mathematics, computation sometimes generates new mathematical theories. But this does not happen in medicine I suppose.

Wiens: No, that's not true. It does happen. These models are hypothesis generation tools, but there's a limit in medicine to the kinds of hypotheses we can test. The gold standard is a randomized controlled trial, but there are many hypotheses for which it would be unethical to run a randomized control trial. If you wanted to look at the effect of pain medication on one's risk of acquiring infectious disease, are you going to deny half of the population at your hospital pain medication? That leads us to causal inference, which I think is a really interesting topic as well. We certainly do generate hypotheses, but we want an even higher level of interpretability. It's not enough for the model to just produce explanations. Those explanations have to be reasonable, at least in part. In medicine, we're far from knowing everything, but there are some things we do know, and the model should agree with this knowledge.

Wang: From the discussion, I feel that in the machine learning world, or even in the traditional design automation world, people try to solve problems because they feel they can solve all the problems. But at a certain point we should try to understand that a problem can become unsolvable in practice, because if we don't know the boundary of our technology, we're going to invest so much on those practically unsolvable prob-

lems and waste resources. A lot of times the problem may not be solvable in a particular way, but if we understand that it's not solvable we can always work around it. For example, if we know that the neuromorphic chips can only achieve 98% accuracy, we need to design a controller to assume that. But today, people don't know how to characterize the boundary, and that's making everything harder because everybody has a different view, making different assumptions when they put things together. It can become a real mess. We need to look not only with a problem-solving mindset by implicitly assuming everything is solvable but also to understand the limitations in certain scenarios and realize that certain problems as they are formulated aren't solvable in practice. If we achieve a good understanding of that, we can find a way to work around it.

Basu: Well, we have discussed so much at this point that we may actually be at the start of a huge new area of scientific endeavor. So much is not yet known about autonomous machine learning systems as yet, and the future possibilities are most intriguing. And with that, this roundtable must come to a close. Thank you, all, for your time and comments in making this roundtable so productive. ■

Resources

[1] S. A. Seshia, X. J. Zhu, A. Krause, and S. Jha, eds., Report on Dagstuhl Seminar 17351 "Machine learning and formal methods," 2017. [Online]. Available: http://dx.doi.org/10.4230/DagRep.7.8.55

[2] S. A. Seshia, D. Sadigh, and S. S. Sastry, "Toward verified artificial intelligence," 2017. [Online]. Available: https://arxiv.org/abs/1606.08514

[3] S. A. Seshia, "Combining induction, deduction, and structure for verification and synthesis," *Proc. IEEE*, vol. 103, no. 11, pp. 2036–2051, 2015.

About the participants:

Sankar Basu *is a permanent member of the National Science Foundation scientific staff and a program director.*

Ruchir Puri *is a scientist and the chief architect of IBM Watson and an IBM Fellow.*

Sanjit A. Seshia *is a professor in the Department of Electrical Engineering and Computer Sciences at the University of California, Berkeley.*

Jenna Wiens *is an assistant professor of computer science and engineering at the University of Michigan.*

Li-C. Wang *is a professor and director of computer engineering at the University of California, Santa Barbara.*

David Yeh *is a Texas Instruments assignee at SRC, where he is a senior director. He is also Roundtables editor of* IEEE Design&Test.

■ Direct questions and comments about this article to David Yeh; David.Yeh@src.org.