Embracing empiricism - from the lottery hypothesis to creating real-world impact: an interview with Jonathan Frankle

DOI: 10.1145/3811926.3811930

Jonathan Frankle is Chief AI Scientist at Databricks, where he leads projects on the AI research team to bring the latest advances in artificial intelligence and reinforcement learning to Databricks' tens of thousands of enterprise customers. He arrived via Databricks' $1.3B acquisition of MosaicML, which he co-founded as Chief Scientist. He earned his PhD at MIT, where he empirically studied deep learning with Prof. Michael Carbin, specifically the properties of sparse networks that allow them to train effectively (his "Lottery Ticket Hypothesis" - ICLR Best Paper, ACM SIGAI Dissertation Award, MIT Dissertation Award). He earned his BSE and MSE in computer science at Princeton.

Our Editor-in-Chief, Ella Scallan, sat down with Jonathan Frankle to discuss the lottery ticket hypothesis, for which he was awarded the 2023 AAAI/ACM Doctoral Dissertation Award. In this wide-ranging conversation, Jonathan delves into empiricism vs theoretical proofs, how the approach to computer science has changed (even if the fundamental problems haven't), how younger researchers are rapidly adapting to a world that values impact above all else, and what it means to be a researcher. Read on for an insightful and thought-provoking discussion.

You were awarded the 2023 AAAI/ACM Doctoral Dissertation Award. What was the topic of your dissertation research, and why was this an interesting area of study to you?

It was on this topic that I called the lottery ticket hypothesis, which I first developed back in 2018. The goal was to understand how deep neural networks learn, why they learn, and what they learn. Despite all the attention around neural networks in the past five to ten years, it's still a very mysterious process and I don't think we've gotten any closer to a clear answer. In fact, any answers we've achieved have been out of date by the time we've achieved them, because things keep changing - the systems keep becoming bigger and more complex.

One question I asked was: how big do these systems have to be to learn? Generally, we make the models bigger and they learn better - but is this necessary? There's this strange phenomenon that's been observed in the literature for decades - and was especially popular a few years ago - where after you've trained a deep neural network, you can actually delete a lot of the parameters and the models perform just as well.

This is weird because it makes you ask the question - did the model have to be that big in the first place? But if you were to take that smaller neural network with all those connections deleted and try to train it from the beginning, it doesn't learn very well. So, why does it seem like whatever the neural network learns by the end can be relatively small, but the learning process requires it to be big? Intuitively, that could make sense. For instance, it's harder to wrap your head around a complex topic than it is to have a nice synthesized understanding of that topic.

I found this strange answer that applied to what were meaningfully sized neural networks then. Specifically, when you create a neural network and you create all these connections, you set each connection to a random value at the beginning, just by sampling from some distribution. These random values have some properties in the aggregate, but the specific values seemed unimportant. However, it turns out that after you delete the connections in the final neural network, the specific values of the remaining connections actually matter a lot. If you train a smaller network using only those connections with those specific values, it can learn. If you train a smaller network using new values sampled from the same random distribution as the original network, it doesn't learn. This is very strange. So you could actually train those smaller neural networks from the beginning or near the beginning of training, so long as you set those parameters properly.

To some extent, my thesis was trying to tackle this question of how big a neural network needs to be in order to learn, when what you learn seems smaller than what is needed. Then I found this weird property that maybe you can learn with something smaller, and that there is something special about the random values. This is where the lottery metaphor comes from. It's a small insight into how neural networks might learn, which continues to elude us. But any insight is important.

What was the impact of your work?

The main finding was the observation that maybe there isn't a difference between the amount of capacity you need to learn and the amount of capacity you need to know something in a neural network. And that perhaps there is some method to the madness in how these things learn. From a technical perspective, it was the insight that the sub-network you get after training and pruning could have been trained from the beginning, if only you knew the right parameters.

The theme from my work that has stood the test of time is the focus on empiricism. This sounds silly in 2026 - like of course we do empirical work to understand AI. What else would you do? In 2018, it was actually pretty strange and controversial to not have some nice elegant theorem to tell us what to do. People were hoping for some grand unified theory, because I think we do have a bias towards that in computer science, as well as towards properties that guarantee certain behaviours.

It was quite uncomfortable when I talked about this work at that stage, as I had a hypothesis about how learning happened in deep neural networks, but I couldn't show mathematically that this had to be true. That was actually a big challenge. Some people really didn't like the work as a result, including one of the reviewers of the paper. When I was at the faculty market, someone asked me this question which is burned in my mind: "You've done this empirical work. Do you plan to do anything principled?" I knew what they meant: principled in

computer science is code for formal mathematics. They wanted to know if I was going to prove my work, or if it was 'just' empirical. But I resented the subtext that empiricism is unprincipled.

I think principled empiricism is a very valid way of getting knowledge about the world. In any science other than computer science, that is our entire way of getting knowledge about the world. That was a somewhat controversial point of view in computer science and AI at the time. It feels very silly looking back, because now everything is empirical and we have no other approach - the sheer scale and complexity of the systems is such that it defies any mathematical framework we have. And real data defies mathematical description - if you're training a model on all of human language, how do you mathematically describe the internet and all the data on the internet? In 2017 and 2018, there was a large community of us who did believe in the empirical approach, but it was such a weird minority view that I don't think was very welcome in traditional computer science circles.

My dissertation was so unapologetically empirical among a lot of other work that was unapologetically empirical at that time. In some sense it became a bit of a symbol of that - people saw this well-known paper that had shown a really interesting piece of scientific insight, purely empirically. I think that was part of a trend of folks who helped to change the way we view doing science and AI. To me, that's like the lasting impact of the work, much more than any specific insight on the size of neural networks.

It sounds like there's been a real mindset shift in AI. So, in 2017/2018 you were finding the parameter values via brute force. Has anyone come up with an alternative methodology since then?

I haven't seen anything. People still propose methods with reasonable frequency, although when I see papers on this topic, I generally tell people to find something more relevant to work on. It was an interesting paper for 2018, but it's 2026 now. The nature of what science is worth doing has changed a lot, and I don't think this is a problem worth working on today.

In some sense, the original method in the paper has stood the test of time because it is so simple. It works like this: take a big neural network, save the values of every weight that was assigned at the very beginning, train that network to the end, delete whatever parts end up being unimportant at the end of training, and then just imagine you had known that at the beginning of time and go back. You can't find the smaller network unless you've trained the bigger one, hence why this method is reasonably impractical for any real applications. And the best way to do this is actually to do this incrementally, by getting rid of a few parameters at a time, and iterating this process a dozen or a couple of dozen times. You can see how this becomes expensive fast and your advisor wonders what in the world you're doing. So that is the gold standard method. There are lots of other methods in the literature, some of which are more efficient, but with lower quality.

Thanks for clarifying. I can see how that negates the potential benefit of having a faster network.

At the time, the paper raised the hope that you might be able to find this smaller network more efficiently, now that we know it exists. But it has been hard to find. It's also that the

nature of the smaller network that you get is one that's relatively hard to take advantage of on contemporary hardware. Because our hardware is designed to do these big blocks of computation where you have a matrix that is completely full of numbers. And if you're missing a number here and a number there, it's hard to take advantage of that.

As you said - that was 2018, this is 2026. What are you focusing on now? What does your research look like these days?

For me, the most fundamental problem in AI right now is not how smart it is or how efficient it is, but how we know if it's working. I think we lack good science on that topic at the moment. The nice thing about this problem is it's one that doesn't require GPUs, a supercomputer, or tens of millions of dollars. So for my team at Databricks, there's no excuse that we don't have the money that OpenAI has or Google has. The only thing you need is humans and your own ingenuity and creativity. It's a really lovely problem from that perspective.

I almost feel like I'm an HCI researcher these days because I have one unfair advantage that I think a lot of my colleagues in academia don't have. I have real users, I have customers, and I can just go talk to them and ask why they're using AI and why they're not using AI and where they're stuck. In the academic world we rely on these benchmarks. There's things like SWEbench, Humanity's Last Exam, or Math Olympiad problems - things that have even made it into the popular press as measures of AI progress. Those are all supposed to be proxies for how valuable this technology is in the real world. My measure is to go into the real world and ask the users directly.

Using these insights, I do a lot of fundamental research on what's necessary to make this work better in those real settings. So I'm doing quite a bit of reinforcement learning these days, which turns out to work reasonably well for solving these problems. I spend a lot of time trying to figure out how to build better benchmarks.

It sounds like a key problem to solve as AI systems become more embedded into people's lives.

Yes, and it's very hard for computer scientists to grasp, because now we're not just going from science in the theoretical and formal world to science in the empirical world - we're getting to the question of what makes systems good for humans. And so that is another conceptual leap for computer scientists that may take some time to sink in. We're moving closer to the boundary where we leave mathematics behind and get closer to the real world. Each of those leaps tends to take time for computer science as a scientific community to get around.

It's great to hear that you're working on this aspect of AI, because I think it's been neglected. So much of the focus is on building bigger, on the Silicon Valley ethos of 'move fast and break things', without really accounting for what makes human life better. It's an important question, when the level of technology we have already far outstrips how much we can use it.

I would frame it even a little bit differently. If you are a capitalist and all you want is to make the most money, I don't think the bottleneck to that is how smart the models are. I think AI is smart enough to solve a much wider range of problems than we have yet used it to solve. In

my view, the limiting factor is that we wouldn't even know if AI had solved those problems because we don't know what it means to solve them. It's very hard to build a system or get a system to do something well if you don't know what success looks like.

In the old world of coding, there was no ambiguity about what a program is and is not supposed to do. You got a clear specification, you could write tests and measure that. It's almost second nature to us at this point given the progress in software engineering over the past half century. That is much more difficult for AI because the nature of the tasks is more nebulous and fuzzy. I think that computer science as a field has a choice of whether to be left in the dust or whether to change with the times and embrace ambiguity and embrace fuzziness and embrace something even more empirical and even more imprecise than we've had to embrace in AI over the past half a decade.

This is where I'm more rude to my computer science colleagues. I say that computer science may become the new electrical engineering, which computer science emerged from as it embraced the boundaries of hardware into software and algorithms. We went far beyond building electrical systems and blossomed into this extraordinary field. I think if computer scientists would like to live in our nice, neat, discrete, specific, measurable, formal mathematics world, that is not the real world and that is not what computing looks like today. And we have the choice of whether to embrace the continued evolution of computing, or to be left behind and let other fields take up that banner. I don't really know if I'm a computer scientist anymore in the formal sense, because I don't really know whether my field looks at the work I do as being computer science.

On this basis, every time I do an academic visit I try to rile up whichever department I'm at, because I think that taking a bunch of algorithms and systems classes does not remotely prepare you for the kinds of problems that actually matter for computing today. And nobody taught me empirical research. Nobody taught me how to run experiments involving humans during my Bachelor's, Master's, or PhD. And yet, hypothesis testing and experimental design are some of the most important skills today. I didn't even have to take a stats class as part of any of my computer science degrees. And I think that we need to get with the times.

We've touched on this already, but which questions in computing do you think are important at the moment?

I think this is one of my favorite things about computer science. The problems are always the same - they just show up in new and different forms. Fundamentally, I don't think there has been anything new in computer science for decades. That's not meant to denigrate the field. It's to say that we keep seeing old problems reflected in new ways and new contexts.

My favorite personal example of this is this measurement problem for AI. How do we know if AI is good? I think of this as a problem of programming AI. In the same way that you program a computer by explicitly telling it what to do, and then verifying whether it did that. It just so happens that with AI, there's no code to look at. All you do is you write some test cases and then ask the computer to do a better job of meeting those test cases. And then you check whether it did that, and if the computer does meet your test cases but still doesn't do what you want, you have to update your test cases because maybe you've missed something. It's different, but it's programming.

To bring this back to the ACM perspective on problems in computer science - when you go back and read the literature from the 1960s, it resonates just as much today as it must have done then. There are so many beautiful quotes from that era. Here's one from Edgar Dijkstra, who's very quotable. “In those days, one often encountered the naive expectation that once more powerful machines were available, programming would no longer be a problem.” Today, that means that if the AI is smart enough, we won't need to worry about measuring it or figuring out whether you want it. There's nothing new under the sun in computer science and it's why we, as a field, have so much to offer as we try to make sense of AI. If only we're willing to meet the moment and adapt our frameworks and our epistemology to this new kind of computer that we're trying to understand and program.

This makes me think of another one of my favorite quotes. There was a NATO conference in 1968 on the software crisis - we didn't know how to program computers. The quote is from that report: “certain classes of systems are placing demands on us, which are beyond our capabilities and our theories and methods of design and production at this time.”

The concerns haven’t changed then - just the context.

That's what I love about computer science. It is the same problems in a new light. It's a question of specification, programming and verifying the specification. That is every computer system we have ever tried to get to do what we want. That is what programming is. How that manifests in AI, how we put that into practice, fill in the details, and build a methodology will be the work of the next half century.

And rather than lowering our standard of truths, we need to allow for fuzzy truths rather than just formal mathematical truths.

Yes, exactly. I think there's a perception in computer science that formal mathematical proof is the highest form of truth. And there are lower truths, like empiricism or data, and even lower truths, like doing human studies. There is this hierarchy of truth. I think this is reflected in faculty hiring, and in many ways in which work is valued and rewarded in the field. I think it has changed a lot.

For example, the fact that we're having this conversation and that a dissertation was awarded in 2025 for work that was actually done in 2018 says that we've come a long way in our embrace of empiricism. This award was issued a decade after the work was actually done, given some of the time lags of my career in the pandemic.

However, I think there's always a reluctance. And in some sense, the folks who are most senior in our field right now are folks who came of age in a very different era. We have to be willing to adapt our frameworks to meet the complexity of the problems. It's hard to have formal mathematical proof about things that involve humans. That's kind of intrinsic to the world. It's also hard to have formal mathematical truths about AI systems. They are so unfathomably complex that they defy easy characterization in math. We have to adapt our approach and adapt what we consider to be knowledge so that we can make progress, accepting that our knowledge will be intrinsically incomplete.

That’s embracing reality, right. If an AI system is incredibly complex and impossible to formalise, well, so is the real world.

And that's not to denigrate this approach. It's given us our entire theory of computing, on which we built our digital world. It's given us cryptography and security. It's given us mathematics. These are incredibly valuable ways of attaining scientific knowledge where we can afford to use them. But if we're too precious in demanding that standard everywhere, it will mean that we won't make any scientific progress in the key problems of our time. And so I do get in some trouble every time I do an academic visit and talk about this - typically with the older faculty. Generally, it's what gets the students most excited, which makes me very excited for the future.

Do you think we'll continue to see this trend of embracing empiricism?

I think it's going to show up in more parts of computer science, especially as computer science becomes more relevant to the world and more a reflection of the world. Students will get to vote with their feet. The beautiful thing about older generations is that they go away, and younger generations who have different perspectives show up. When I look at the generation of students who've come up behind me, they have an even greater embrace of both the messiness of these problems and the promise of this technology in a way that goes beyond my imagination. That's the joy of being an academic and of being a scientist who trains other scientists. You get replaced by people who have even greater creativity and stand on your shoulders and do even greater things.

So while I complain about the people who are stuck in their formal world, the students coming up behind me complain that I'm stuck in this world where I want to write code with my hands and not let AI do it. And that I'm not focused enough on the messiness - that I'm too focused on science and measurement, and not enough on building and just creating things in the world and letting adoption be the measure of success. That is an even more empirical, messy, pragmatic embrace of the complexity of the world. In some sense, I'm becoming a dinosaur very quickly! And that's great. That's the joy of science. We make progress and older dogma that doesn't adapt gets replaced - me included.

It's exciting, but I think that our field could move faster to embrace these problems and support young scientists who are focusing on these problems and give them the career opportunities to really thrive and embrace these new ways of looking at the world.

It sounds like you're learning a lot from younger researchers and it's great that they have so much vision. Do you have any advice for these early career researchers in your field?

My advice is aligned with what I've said: build.

The mode shift I've seen in science lately has been towards real systems that don't just show something that might be true in the world, but systems that make that become true in the world. And I love it. I think there's something just so beautiful about going to the ultimate demonstration that this was valuable science in the world - even skipping out on the measurement and going straight to the impact. There's just something very inspiring about that approach, and AI makes it a lot easier. I'm a slow person because I stop and ask how we know if it's good. The answer is, well, who cares about trying to figure that out? The way we know it's good is because other people think it's good and use it. It's almost a little bit uncomfortable for me. I can only imagine how it feels to someone who is doing the theory of

computer science to see a faculty candidate for whom their CV is that they've built a bunch of things that people have adopted, and maybe haven't published any papers. To me, that is the wave of the future. And it's so delightful that the laboratory for our field of science is the real world and adoption. That's thrilling.

That sounds really like a real paradigm shift. I'm excited to see what happens in the years ahead.

I would push back on that. I don't think it's a paradigm shift - I think it's a throwback to the 1970s when we had BSD, Unix, C, and a lot of what we would consider the great accomplishments of our field of that era. People adopted and used TCP IP. There's no math to show that that's an optimal protocol or that's an efficient protocol. It just worked. So in some sense, we're back in a world where we have this new substrate of computing and we're limited by our imaginations of what to do with that. And we'll fill in the gaps and we'll understand the formality later. For now we get to create.

I like that worldview because it feels like we get a little lost in measurement and box ticking in the modern world. My final question is - were you always set on being a researcher?

No, definitely not. Am I a researcher now? Or am I just a capitalist who builds products and sells things? I think that's in the eye of the beholder, but I'd like to think that my primary contribution is still new human knowledge. But it was very late in my undergraduate career when I even contemplated doing a PhD at all or doing research at all. And it was very late in my PhD when I decided I wanted to become a professor and then, that never happened - I stayed in industry and enjoyed the startup world. So, I don't know if I want to be a researcher. I still haven't figured that one out. I guess I do research these days, but what does it even mean to be a researcher in a world where the primary currency is now building rather than writing papers? The best research, again, is adoption. Is that really research? I think that all depends on perspective.

I want to do useful things in the world with my technical skills and my scientific skills. And if that makes me a researcher because I'm using scientific skills and trying to create new knowledge or new artifacts, great. If I'm a boring manager in industry and not a researcher, again, that's in the eye of the beholder. I suppose it comes back to the question of whether people value empirical research and real-world impact or not. Would you rather have a beautiful proof or would you rather have 10 million users who have built on what you've done? And which one is more valuable for humanity? I certainly have my preferences on how I like to contribute. There are lots of ways to do useful things in the world through science.

Definitely! Thank you, this has been an interesting conversation.