EPISODE 813

[INTRODUCTION]

[00:00:00] JM: Until Google DeepMind came into the field, protein structure prediction was dominated by academics. Protein structure prediction is the process of predicting how a protein will fold by looking at genetic code. Protein structure prediction is a perfect field to approach through the application of deep learning, because the inputs are highly dimensional and there is a plentiful array of different sets of labeled data. Protein structured deep learning is a field in which many different approaches can be taken often involving supervised learning and reinforcement learning.

Mohammed AlQuraishi is a systems biologist at Harvard. His background spans computer engineering, statistics and genetics. In his work, Mohammed explores the interplay between biology and computer systems. One area of Mohammed's focus is protein structure prediction. In a blog post last year, Mohammed gave a brief history of protein structure prediction and described the significance of DeepMind entering the field. DeepMind's AlphaFold technology surpassed all other competitors in the most recent CASP protein structure competition.

Mohammed joins the show to discuss biology, academia, deep learning and DeepMind. Mohammed is wonderful at explaining complex ideas in elegant terms, and I deeply enjoyed the conversation we had together.

Recent updates from Software Engineering Daily land. Podsheets is our open source set of tool for managing podcasts and podcast businesses. We have a new version of Software Daily, our app and ad-free subscription service available at softwaredaily.com, and we're looking for help with Android engineering, QA, machine learning and other roles. You can find our Software Daily FindCollabs in the show notes for this episode.

The FindCollabs hackathon has ended. The winners will probably be announced by the time this episode airs. We will be announcing our next hackathon in a few weeks, so please stay tuned. The links for all these updates are in the show notes.

Let's get on with the show.

[SPONSOR MESSAGE]

[00:02:23] JM: Today's episode of Software Engineering Daily is sponsored by Datadog. With infrastructure monitoring, distributed tracing and, now, logging, Datadog provides end-to-end visibility into the health and performance of modern applications. Datadog's distributed tracing and APM generates detailed flame graphs from real requests enabling you to visualize how requests propagate through your distributed infrastructure.

See which services or calls are generating errors or contributing to overall latency so you can troubleshoot faster or identify opportunities for performance optimization. Start monitoring your applications with a free trial and Datadog will send you a free t-shirt. Go to softwareengineeringdaily.com/datadog and learn more as well as get that free t-shirt. That's softwareengineering.com/datadog.

[INTERVIEW]

[00:03:24] JM: Mohammed AlQuraishi, you a systems biology fellow at Harvard. Welcome to Software Engineering Daily.

[00:03:29] MQ: Thank you very much for having me.

[00:03:30] JM: I'd like to start with some of your background before we get into a discussion of proteins and deep learning. You've studied computer engineering, biology, statistics and genetics. How would you describe your research interest?

[00:03:46] MQ: So I guess I'm a computational biologist. That's what I would call myself. In general in interested in using machine learning and just algorithms in general to try to understand molecular interactions. So how molecules interact in a cell, and then using that to sort of say something about how cells behave and how diseases arise.

[00:04:03] JM: Do you think of biology as a statistical process with some unknown paths, or do you think of it more as a deterministic process?

[00:04:14] MQ: No. I would say it's very a probabilistic process, but it's not entirely random. So it has elements of logic and things that one would associate with computer programs and software, but is sort of imbued with that kind of probabilistic flavor, if that makes sense.

[00:04:29] JM: So it's 2019, what are the areas of biology that have become easier to innovate within things to modern computation?

[00:04:37] MQ: Certainly, genetics and genomics. I mean, those are the big ones where they are very data-driven, and we have just so much data that we could simply leverage computation. But I would like to think as well that the things that are kind of molecular modeling, molecular biology, trying to understand how biological molecules do things. That's definitely benefited in the last decade, and I think we're going to see a lot more changes and improvement in the upcoming few years.

[00:05:00] JM: Are there any other areas that maybe they're not quite ready yet to be approachable through computational biology, but maybe they're around the corner, like there's some cool stuff going on in the lab, some promising experiments. What's around the corner?

[00:05:16] MQ: Yeah, that's an interesting question. People have been talking about systems biology for a long time, and the idea there is that you're not only modeling sort of individual things, but you're modeling systems of things, how many molecules or even how many cells come together to do stuff. I would say that hasn't quite yet reached the level of maturity to sort of be useful broadly, but I think that's – At the moment, I would say are still outside the reach of computation. But I think that may well be on the corner 5 to 10 years.

[00:05:46] JM: One thing that strikes me as being hard about biology – I mean, biology is like computer science and that we're looking at these large complex systems with these different levels of abstraction and these different high-level abstractions that get composed together in useful ways. But the stark difference between biology and computer science is that wit computer science or software architecture, we can abstract away from the areas that we're less sure of,

like physics. We kind of have these low-level corrective mechanisms to bend physics into being in states like 1 and 0. At a lower level, obviously it's a much more of a granular system, but at the level that we get to operate at, it's the 1 or 0 level, which is such a profound breakthrough in technology.

But in biology, again, there's so much we don't know. It is this analog system in some sense, and there are so many different pathways and chemical relationships that we don't understand. As a systems biologist, how can you gain any confidence in your theories when there are so many confounding variables?

[00:07:01] MQ: Yeah, there's a lot to unpack there. A few things to say, in some ways, genetic information is digital. So at least in the level of genetics, sort of life has figured out a way to encode things using four bits instead of two bits with ACGT, which is interesting. I mean, in priority, maybe that wasn't obvious that that would be the case, and I think in some ways that's what drew me to Biology, actually is, "Oh! This is digital too. It looks a lot like computer science." But you're quite right, and that it's different in many other ways including the fact that at least aspects of the computation done by biology is analog. Maybe more –

To me, personally, what I think is actually the biggest difference between computer science and biology is the fact that the former is human engineered, so there are human artifacts, while biology is not. So the kind of design principles or the intuition that we have how abstractions are to be built in computer science don't really show up at all, or at least they show up in a very different way in biology, because they all sort of evolve organically. They have this set of trajectory that that's very kind of accumulative and iterative. They're quite different than computer science or than just software programs. That to me is really the biggest gap between those two, the two things, or the two worlds.

[00:08:14] JM: So when I was in college, I studied biology for a little bit and then I switched to computer science. In that transition point, there were a couple of books that I looked at. I didn't make through either of them in their entirety. I read probably 5 to 15 pages in each of these books and kind of skimmed them, but it stood out to me as some kind of gap bridges between the fields of computer science and biology. One of them was the Selfish Gene, which takes this kind of bottoms up or bottom to top evolutionary kind of discussion of soft of how the primordial

soup might have be turned into, first, the lower level abstractions that underpin biology and then eventually led to these higher level organisms that we have. Then the other book was this Titanic book, the G*ö*del, Escher, Bach book, which is it's interesting in the sense that it talks about these idea of lower level languages and does bridge some gap between the ATCG and the 1 and 0 world, but I found that book to be a little bit too – It was hard for me to, I guess, get into such detail and tries to draw so many different analogies that it becomes really hard for me to really make sense of what arguments he's trying to make. At least that's my 5 to 15-page amateurish critique of it. But can you go a little bit deeper on the bridge between computer science and biology and what you see as perhaps the parallels between the two worlds.

[00:09:54] MQ: Yeah, sure. I mean, by the way, these are both great books. Actually, to me they were very influential. I got them when I was in college or even earlier in high school, and they definitely had a very big influence on my thinking. So I think they're both great books.

G*ö*del, Escher, Bach, incidentally, I think is really more focused on the question of consciousness instead of maybe newer science and a little bit less about biology wildly construed, but that's beside the point.

Yeah. In some ways, I think maybe the most fundamental unity between computer science and biology is the fact that they're both trying to or they both have built programs. So this notion of you want to build some sort of reactive kind of object that's doing something, that has some purpose, that's able to interact with this environment. In our case, in the computer science case, it's a user or another program interacting with it. In biology, it's either organisms and it's where it lives. That's the environment. But it's able to interact with that environment to do stuff and to – In the case of biology, to survive and procreate and so on. So they're both trying to execute some sort of logic. That kind of basic premise I think is really the thing that unifies those two sort of very distinct kind of classes of phenomenon.

So far as one's trying to do this both for human engineer designing a program for a life form evolving over eons. To be successful in that, the idea of abstraction, the idea of hierarchy, or sort of building things from more primitive components and then building layers upon layers. That approach is very useful, because it allows – Well, it's interesting. From the perspective of

the human design, it allows one to reason about these systems to do reuse, and that sort of thing, kind of modular architectures.

In biology, you don't have this notion of reasoning over the system that we do in a human engineered program, but you definitely have this notion of reuse, where instead of biological systems become more evolvable, if they are using abstractions and if they are using kind of these primitives built out of smaller primitives. In that case, there are actually parallels. This is at least kind of one aspect of your question.

[00:12:04] JM: All right. Well let's get begin to move from the philosophical into the concrete. I'd like to give a little bit of a biological review for software engineers who never took a look at biology or who are a little bit foggy in their recollections, because we need to talk about what a protein is. Can you explain what role proteins play in our body and kind of what a protein is at a fundamental level?

[00:12:29] MQ: Sure. So one way of looking at proteins is that they are nano machines. They are these small molecular machines that do stuff, and they do many things. They do mechanical things. They do structural things. They do informational processing things. So on the mechanical side, for example, they actually – They're these sort of cargo walkers that transport things across the cell, and that transport process is done using proteins. They're also information processors. They are proteins that sit on the surface of your cells and they sense what's going on inside and outside of the cell. Based on that, they execute new commands effectively, so transmitting information. So they're really kind of the workhorses, the things that actually do stuff in your cell.

[00:13:11] JM: One thing I think is interesting, kind of the parallel we could draw between proteins, perhaps proteins and programs, or proteins and modules, or proteins and applications versus the code that leads to those applications. In the protein world, you have the ATCG in the case of DNA, but not only is it – Then you have the ones and zeroes. Obviously, they compile into these grand – Well, I guess these grandiose program text gets compiled down into ones and zeroes. So it kind of have the code to the ones and zeroes. Then the ones and zeroes get run.

But in the cell, what's cool is you kind of have these – You have the applications running, which are the proteins, then there's also like – Concurrently, there's also genetic code that's just like floating around and then there's other proteins that are on the genetic code kind of acting as Turing machines on that genetic code and producing more proteins. So within the cellular structure, it's like you have continuous compilation alongside the computation, which I think is a little bit different than at least our current computation models.

[00:14:25] MQ: Yeah, absolutely. It's not a normal machine. I mean, that's what's really cool about it, is that the information substrate and kind of the actual execution, they're all sort of mangled up, because they're all molecules at the end of the day. You have information being transmuted into machines, and then these machines sort of directly operate on the information, and so it's all entangled. It's quite different architecturally.

Just parenthetically, as a computer scientist, when I went into biology, this was for me sort of the most difficult paradigmatic shift I had to make, because I really came to it from someone like a computer engineer, somebody who thinks in terms of memory, and disk, and logic, and so on, and CPUs, and this was just so different.

[00:15:08] JM: So these proteins are made inside of our bodies. How well do we understand the biological pathways of protein creation?

[00:15:19] MQ: On one level we understand it very well in the sense that we know that proteins are specified by genes. So these are contagious segments, or even non-contagious segments of DNA, and there's sort of a mapping, a code, that takes you from basically triples of DNA basis, like the AGs and Ts to a different language, if you way, a 20-letter alphabet that makes up proteins, and proteins are chains made up of this 20 amino acid, or 20-letter alphabet. In that sense, we understand it very well.

What we don't understand, once you have this sequence, this arrangement of the 20-letter alphabet for a given protein, we don't know what the resulting protein will look like. Often, the shape of the protein is the same as what it does. We have this, in some sense, kind of shallow understanding. We kind of know the mapping of what makes up a protein, but we don't really understand how that makeup translates into function and structure.

[00:16:12] JM: So perhaps to draw towards your analogy, we don't really understand how the compiler works.

[00:16:18] MQ: Yeah, in a sense. That's reasonable. I mean, that's one way of looking at it. Sure.

[00:16:21] JM: Okay. We'll talk a little bit more about that. So in ways is that analogy wrong? We've got this genetic code that's floating around. We've got these finished proteins that result from that genetic code. Why is the process from getting to genetic code to the finished protein so opaque?

[00:16:39] MQ: Actually, there's another wrinkle here, which I think is worth mentioning with respect to how this is different from computer science, is that you actually don't have a compiler, for the most part. This is all driven by self-assembly. So it's all driven by the laws of physics. You don't have something that's sits in and says, "I'm going to –" I mean, you sort of do, but it's not in the same way that compilers will going to sense that you're doing this sort of – You take a protein sequencing, you fold it. Instead, it's just basically a chemical process that that gives you the protein.

But the challenge becomes, is that now that we have this protein, and in its initial state, it's effectively the un-compile state. Basically, it's like a thread. It's a chain in 3D space. It doesn't have a shape, a well-defined shape. But then just going through the motions of physics, that thread takes on a well-defined three dimensional shape. You could think of that as like the compilation, but the compiler is just physics.

The challenge here is that that process is very complex. There are very, very many possibilities that that could arise in terms of how you go from that initial shape to the final shape, and it's not something that we have sort of fully – We don't fully understand. Actually, if we want to simulate it, it's computationally very intensive. So there's that too.

[SPONSOR MESSAGE]

[00:18:04] JM: DigitalOcean is a reliable, easy to use cloud provider. I've used DigitalOcean for years whenever I want to get an application off the ground quickly, and I've always loved the focus on user experience, the great documentation and the simple user interface. More and more people are finding out about DigitalOcean and realizing that DigitalOcean is perfect for their application workloads.

This year, DigitalOcean is making that even easier with new node types. A $15 flexible droplet that can mix and match different configurations of CPU and RAM to get the perfect amount of resources for your application. There are also CPU optimized droplets, perfect for highly active frontend servers or CICD workloads, and running on the cloud can get expensive, which is why DigitalOcean makes it easy to choose the right size instance. The prices on standard instances have gone down too. You can check out all their new deals by going to do.co/sedaily, and as a bonus to our listeners, you will get $100 in credit to use over 60 days. That's a lot of money to experiment with. You can make a hundred dollars go pretty far on DigitalOcean. You can use the credit for hosting, or infrastructure, and that includes load balancers, object storage. DigitalOcean Spaces is a great new product that provides object storage, of course, computation.

Get your free $100 credit at do.co/sedaily, and thanks to DigitalOcean for being a sponsor. The cofounder of DigitalOcean, Moisey Uretsky, was one of the first people I interviewed, and his interview was really inspirational for me. So I've always thought of DigitalOcean as a pretty inspirational company. So thank you, DigitalOcean.

[INTERVIEW CONTINUED]

[00:20:12] JM: So just to go a little bit into the physics, and by the way, I failed physics in high school. So this may be really tortured, but I think like it as it kind of works, and I think I got a C in chemistry. But you have DNA, which is deoxyribonucleic acid, and that's like these amino acids, the ATCG, and they're situated in these base pair.

[00:20:42] MQ: So those are the base. Their amino acids are different, the amino acids that would make up a protein.

[00:20:47] JM: Oh, sorry. Oh, okay! Right. So what are ACTGs again?

[00:20:52] MQ: They make up the DNA, right? The ATCGs make up the DNA, and the amino acids make up the proteins. Okay. There's actually four DNA letters, if you may, and there are 20 protein letters.

[00:21:06] JM: Okay. Actually, I shouldn't be the one explaining this. Maybe you could just give me a little bit more of the outline of the structure of this DNA string, or I think you described it as a thread that's like floating around and it's like – Let's say this DNA thread has been manifested in the cell somehow, like some other protein that perhaps produced it or it's gotten created somehow. I don't remember how it gets created. Then you've got this thread, and this thread has different physical forces pushing on it in different directions that cause it to fold into something useful. Can you just describe it in a little bit more detail those pressures on the folding.

[00:21:45] MQ: Sure. So let me just step back for a moment.

[00:21:47] JM: Sure, please.

[00:21:48] MQ: Yeah. There are, in some ways, two threads here. So one thread is the DNA thread, if you may, and that's – The full size of it, the full thing, is our genome. There's actually multiple in like the human genome, but basically you could think of our genome as one long thread of DNA. That's one particular chemical structure and it's sort of made out of these four different letters that I repeated in a non-aperiodic way. That's one piece. That's, in fact, like an instruction manual that describes the entirety of our genome.

Now, there are small stretches of that DNA that sort of "code" for proteins. A small stretch of DNA effectively gets rid off and then used as a basis to make a protein. So now the protein is a different thread. It's a different kind of thread. The chemistry is completely different. In particular it's made out of those – It's made out of chain of one of 20 possible letters, and this chain is – Well, I mean, the letters are repeated in a non-aperiodic way. So different proteins will have different sequences of these amino acids.

Does that make sense so far?

[00:22:57] JM: It does, yes.

[00:22:58] MQ: Okay. So the DNA has an interesting structure, but that's not really sort of our concern today, and somewhat maybe it's that interesting, although some people will hate me for saying that. Proteins have a lot more interesting structures. The forces that operate in proteins are sort of much richer in a way. The question of structure really kind of into play on proteins and so much on DNA.

So there are forces that play a role. One of the most important is what's known as sort of hydrophobicity, and that has to do – So hydro is water. Phobicity is like fear of. So it's effectively in our cells, they have a lot of water in them. Many of the forces that drive how the protein falls have to do with whether any given part of protein likes to be next to water or it doesn't. If it likes to be next to water, then it's probably going to end up being on the surface. If it doesn't like to be next to water, it's going to be buried inside the protein. So that's at least one type of force that plays a large role in how proteins end up folding.

[00:23:58] JM: Okay. Let's start to move towards the discussion of something that's related to software engineering. There is this field of protein structure prediction. Protein structure prediction is the process of predicting the structure of a protein from its amino acid sequence. Explain what it means to predict a protein structure in a little more detail.

[00:24:21] MQ: Right. So we think that for most proteins, they basically have a single structure. What that means is that thread effectively gets kind of folded up in one particular configuration. That's not entirely true. These things move a little bit, and actually some proteins do have multiple configurations. But for our purposes, we can just imagine there's one configuration.

So the prediction has to do with basically saying where each of those amino acids is going to be relative to every other amino acid. Where are they going to be placed when the protein sort of finally finds its lower energy state, when it finally finds its fold?

[00:24:57] JM: Okay. Basically, because you've got this thread where there's hydrophobic forces that act on the thread in a physically consistent fashion, like every time it seems to happen the same way, because physics doesn't really change for moment to moment as far as we know. These hydrophobic forces are consistent. The relationship is always you've just got this thread sitting in water and the water molecules are – Since they're so pervasive around this thread, you're kind of always going to get the same reaction between all these water molecules that are floating around the non-folded protein structure and the non-folded protein structure itself.

[00:25:41] MQ: That's right.

[00:25:41] JM: Okay. Why do we want to predict the structure of a protein?

[00:25:47] MQ: So the short answer is because the structure determines the function. So if you could predict the — to a large extent. So if we could predict the functions of all proteins, then we would get a lot insight into how a cell works, for example. That's like kind of the basic science answer. There are many applied answers, maybe the important of which is that many drugs operate by sort of modulating the function of a specific protein, typically inhibiting that protein. So if you want to make a drug that modulates a protein, it helps tremendously to actually know what the protein looks like.

[00:26:18] JM: Can you just walk me through what an example protein prediction process would look like?

[00:26:25] MQ: Right. Yeah, that's a good question. So there are many answers because we have many approaches. But I would say kind of maybe the canonical approach that people have been doing for maybe the last 10, 15 years, basically involve this kind of fairly complicated process, millions of lines of code with code to actually do all these. Basically, you take that initial sequence and you try to find correlations in the data that we already have about real proteins that tell you how small fragments of the protein, of how small fragments of any protein look like. So you do this sort of local machine where you say, "This triplet or this 10 amino acid stretch is going to have this shape, because I know I've this shape somewhere else, roughly speaking.

You build this big database, you have all these little fragments and then you do something like a physical simulation where you're trying these different fragments from your database, while at the same time trying to kind of minimize the energy of your protein based on sort of what's known about the physics. You do this over and over and over again for a long time. The hope is that by the time you're done, you end up with something that looks like the real structure. So that's one version.

[00:27:33] JM: In that version that you just discussed, is that a purely computational process, or are there wet lab procedures that you need to do, physical world procedures you need to do?

[00:27:44] MQ: So the process I described is purely computational. I mean, the objective here is to try to computationally predict the structure of a protein. There are of course [inaudible 00:27:53] ways to do this, and the data I was referring to, the data that sort of forms the basis for doing this procedure comes from experiments. But the procedure itself is purely computational.

[00:28:03] JM: There are these other methods for doing protein structure analysis. There are things like cryo-electron microscopy and X-ray crystallography. These are real-world engagements. Can you, I guess, contrast, or describe how these computational methods compare and can be used in concert perhaps with these physical world procedures?

[00:28:25] MQ: Yeah. I mean, suffice it to say that for the time being at least, the gold standard remains experiments, particularly crystallography, and cryo-EM is sort of a newer technology that it's really slipping through the field, but maybe not quite there as being as accurate as crystallography. So these experiments, they're quite laborious. That's the issue. So they often could take a very longtime to get a protein structure, months to years, and the cost estimates vary, but on average let's say it's probably in order of about $50,000 to get a structure. But that doesn't really illustrate the reality. Because what happens, some structures are just very difficult. So you could spend millions and conceivably never get the structure. Some structures are more readily sort of obtainable. So perhaps they wouldn't cost so much. In general, the very laborious process cost a lot of money doesn't always work. But when it does work, it gives you the truth after some bias, but it gives you something that we think is the truth.

The computational approaches vary. The stuff that I was just describing earlier, these complicated three million lines of code type of things, they typically take maybe on the order of – If you're doing kind of the full pipeline on the order of, say, 10 CPU hours. So, generally speaking, not that bad. This is not very expensive. Of course, the problem is that they're still not very good. So maybe they will quickly, but they don't necessarily give you the right answer.

[00:29:47] JM: And just so we don't lose the lead here before we start to get into the details of the computational stuff, talking about the purely physical world experimentation side of things, the 50 grand deriving a protein structure experimentation process, like cryo-EM, or X-ray crystallography. As far as I know, the process, it looks something like you get some purified DNA and then you basically put this purified DNA into water, and then you watch the protein develop into its fully realized form. Then you use these microscopy or crystallography techniques to understand what the outcome was. Tell me where I'm wrong or any clarifications that would be worth making.

[00:30:34] MQ: The details have been a lot on the approach, that actually these approaches are quite different in terms of how one does them. For example, with the crystallography, let's say, there are multiple steps. Maybe the hardest thing is to actually get the protein to "crystalize". You go out there and you have to – You have to get these proteins. You have to make them first. That's doable. You have to purify them. So you have to get them in sort of in a fairly pure form. But then the trick is that you have to get them to actually form a crystal. That process, for the most part – I'm not an expert in this, but it tends to be sort of a lot of black magic, a lot of trial and errors, plus the different conditions that you have to try to different solutions and so on to get the protein to actually crystalize.

Once it crystalizes, it forms basically like a lattice, right? Once that happens, then we basically use sort of X-rays to diffract through this crystal. By getting the diffraction pattern, we're able to sort of work back and get the actual structure. But the rate limiting step to my understanding at least, it's really the crystallization step. Getting a big enough crystal is the hard part.

[00:31:41] JM: Okay. Sounds brutal. So let's get into the computation side of things. There is this community-wide experiment called CASP, critical assessment of techniques for protein structure prediction. Explain what this organization is and give me a brief history of it.

[00:32:02] MQ: Yeah, sure. This is a very important problem, protein structure predictions. It's been going on for over half a century, and people have been trying to solve it for a very long time. There was a period probably in the 90s maybe where there were a lot of these sort of false alarms where people kind of claimed they solve the problem, made a lot of sort of headlines of this in the scientific community only to realize that in fact they had not. It wasn't to say like fraud or anything. People got a little bit too excited.

I think it was on '94 when this CASP sort of process got formalized, and the idea was to say, "Okay, let's try to get at this very rigorously. So what we're going to do is we're going to have this sort of vinyl competition or assessment where we set aside a small subset of proteins whose structures we already know because someone had gone and crystalized them. But they're not yet publicly available. Broadly speaking, people don't know what they look like. Just a handful of people know what they look like.

What we're going to do is we're going to give the sequences of those proteins to predictors, to people who claim they can predict the structure without giving them the structure. Then we'll have them go at it for a few months. Try to make these predictions. Then once everybody does their thing, then by the end of the year, it typically happens in December, everyone gets together and then compares those prediction structures to the ones we know are true and then see how everyone else had done.

[00:33:28] JM: So why has this been arranged in this competition worldwide? Why is this, I guess, a useful construct for the academic world? I guess, now, for the industrial world, which we will get into, or maybe it's been a part of the industrial world for a while. Give me a little bit more context for how this organization or this competition is useful to the world.

[00:33:50] MQ: Yeah. Well, it's useful precisely, because it allows assessment in a blind fashion. The issue before is when you know the answer, it's easy to pretend like you – People have all sorts of biases. So what was going on evidently, because people had access to the true structure, their predictions are effectively cheated in a way. They sort of knew what they were trying to get to, and so they could get to it by forcing the predictions to be completely blind. It

provides this level of rigor that no one can fool others, or fool themselves, in thinking their predictions are working, because they can't compare to the ground truth.

So it's really about providing that kind of installation between data and – If you think about it in machine learning context, it's providing perfect installation between training sets and test sets, and that obviously happens in machine learning as well, like in ImageNet and so on that drove deep learning and all that. So it's perfectly analogous.

[00:34:47] JM: So this has been going for 13 years. Is that right?

[00:34:50] MQ: 14, I guess. Yeah, just over.

[00:34:52] JM: 14 years. So what are some of the approaches that people take to protein prediction and how has the competitive landscape evolved overtime?

[00:35:01] MQ: Actually, I take it back. It's 24 years. There're been 13 of them, but they happen every two years. So it's really 26 years. Sorry. But, yeah.

[00:35:07] JM: Oh, wow! Okay. All right. So we have an even longer timeline to exploring your answer.

[00:35:11] MQ: Yeah.

[00:35:12] JM: So I'm wondering what the prediction, what the prediction strategies have looked like over the years. How has the competitive landscape evolved in the last 26 years?

[00:35:23] MQ: Yeah. So this is kind of before I was on the scene. But my understanding of the history was that I think the very initial phase was actually – I think a lot of it was physics-driven. So people were trying to really seriously simulate these things, and I don't think that went very far. The first sort of major kind of breakthrough or sort of new approach came out of the Baker Lab, which is kind of maybe the biggest group in the states. That was just kind of fragment-based approach I was referring to, where people were doing the sort of thing where I was mentioning, you have this database and you put fragments in and out and so on. That went on

for a longtime and I would say it's the work course of the field. People were iterating and trying different things. But the basic paradigm I think was health-fixed for a longtime.

Then around let's say 2012, 2011, there's a fairly important breakthrough. I can get into it in detail if you like later on. But basically this notion of coevolution where people looked at how these proteins evolved using sort of that type of information. We're able to make such structure predictions. These predictions didn't impact CASP immediately. It took for a few years for it to sort of kick in. But when it did, it make a big difference.

Then now just the last couple of years, deep learning is sort of sipping through, and that's really maybe kind of the third wave now where we have these sort of deep learning based approaches that are leveraging all these earlier ideas but kind of taking them in a new direction.

[00:36:44] JM: So it sounds like the coevolution point is necessary to understand the deep learning side of things. Would you say that's correct?

[00:36:52] MQ: Yes. So far at least.

[00:36:54] JM: So explain that coevolution innovation that occurred around 2011, 2012.

[00:36:59] MQ: Yeah. So it was very much sort of also was driven by the fact that genome sequencing, so this parallel technology, was getting to the point where we're able to get many, many sequences of different organisms and so on. So now what enables is that for a given protein, you could ask effectively for its cousins. You could ask, "Can you give me a list of 100,000 other proteins." If you're looking at the hemeprotein, say, look at analogous proteins and other organisms that are very similar to this protein, but slightly different. They're evolutionarily related, but there are some distance between them.

So what that allows is building what are called multiple sequence alignments, where you take say 100,000 proteins are related and sort of line them up so that you can compare how each position in this protein sort of changes overtime. The key piece are following, the key pieces that in some cases what we observed is that as one position changes, as one amino acid mutates, it

appears that another amino acid in a different location but in the same protein changes soon after. So they're sort of coupling, where one position appears to respond to another position.

The notion is, is that what happens inside the protein is because these things are in close spatial proximity, they're actually close in 3D space. If one of them – If there's a substation where a small amino acid becomes a big amino acid, then that might suddenly kind of clash with its neighborhood. So if there was a new amino that was also big, suddenly it has to shrink. You have to substitute it to a small amino acid.

So these sort of compensatory mutations indicate proximity in 3D space. So we're taking evolutionary information and we're converting it into 3D information. That allows this kind of 3D coupling of things being close in 3D space. Allows one to sort of provide very strong constraints on what the structure looks like, which then enables us to predict the structure.

[00:38:53] JM: Okay. Let's now talk a little bit about deep learning, and then let's come back to that coevolution point, because I think it's worth exploring deep learning a little bit before we get into why the coevolution point is so useful for deep learning-based protein structure prediction approaches.

Okay, I guess speaking broadly, why is protein structure prediction a good candidate problem for the techniques of deep learning?

[00:39:22] MQ: So on one level, it's a well-defined problem. So you have inputs and you have outputs. It's very much a supervised machine learning problem. The inputs being sequences. The outputs being the structure. Typically, it's a one-to-one map. There's not all the noise and kind of unpredictability that characterizes so many other areas of biology, say. So at least in the kind of the physical sciences, it's the closest thing to something that's like a canonical machine learning problem. You could say that there isn't that much data. Typically, there's probably on the order of about 50,000 proteins that really have to work with in terms of structures. So the data is somewhat limited relative to kind of conventional deep learning application domains. But in general, it's amenable to that kind of analysis.

[00:40:05] JM: So that lack of data, can you talk a little bit more about why this is a lack of data. I guess, just describe a little bit more, like – Okay. So I mean, we've done all these shows about kind of what supervised learning is, but maybe you could give like – Describe how supervised learning works at least I the most naïve strategies of protein structure prediction.

[00:40:26] MQ: Right. So supervised learning, the idea, is that you have some model, some function. It could be a neural network. It could be something else. But that function effectively relate to your inputs, your outputs. So the simplest case, if you're trying to predict whether how much a car costs. The input is maybe the model of the car, which year it's made, whatever that sort of thing, the kind of features it has. Then based on those inputs, you have a number which is the cost of the car.

Then typically, you have these parameters. You sort of weight in the function that you don't fix ahead of time. They're free parameters. What the machine learning or the supervised learning algorithm does, it sort of finds the configuration of those parameters that most closely gives you a good mapping between the inputs and the outputs. So you have some go-tos. You know what the two outputs are and then you're trying to kind of optimize this parameter so the inputs map into the outputs. In the context of proteins, the inputs are the sequences, the outputs are the structure. So you want to have some function that does this mapping.

One reason why in some ways supervised learning is maybe a good kind of paradigm for this, is because it doesn't prove to be the case that there are these sort of patterns, these sequence patterns that seem to correspond to these structural patterns. I mentioned earlier this whole notion of fragments and how these fragments are sort of used to build up proteins. So that already tells you that there is some of decomposition in which proteins, really, like hierarchal subjects with multiple subparts. We know, at least in the context of deep learning it can affect national language processing and image recognition, that that seems to be how deep learning works, is sort of decomposing a problem into small subparts. So there, again, is sort of a nice analogy between kind of the canonical deep learning approaches and protein structure.

[SPONSOR MESSAGE]

[00:42:14] JM: When I'm building a new product, G2i is the company that I call on to help me find a developer who can build the first version of my product. G2i is a hiring platform run by engineers that matches you with React, React Native, GraphQL and mobile engineers who you can trust. Whether you are a new company building your first product, like me, or an established company that wants additional engineering help, G2i has the talent that you need to accomplish your goals.

Go to softwareengineeringdaily.com/g2i to learn more about what G2i has to offer. We've also done several shows with the people who run G2i, Gabe Greenberg, and the rest of his team. These are engineers who know about the React ecosystem, about the mobile ecosystem, about GraphQL, React Native. They know their stuff and they run a great organization.

In my personal experience, G2i has linked me up with experienced engineers that can fit my budget, and the G2i staff are friendly and easy to work with. They know how product development works. They can help you find the perfect engineer for your stack, and you can go to softwareengineeringdaily.com/g2i to learn more about G2i.

Thank you to G2i for being a great supporter of Software Engineering Daily both as listeners and also as people who have contributed code that have helped me out in my projects. So if you want to get some additional help for your engineering projects, go to softwareengineeringdaily.com/g2i.

[INTERVIEW CONTINUED]

[00:44:06] JM: Now let's come back to the coevolution innovation that was made in 2011, 2012. How is that a useful factoid or understanding, scientific understanding, to these deep learning hackers that are trying to figuring out protein structure prediction?

[00:44:25] MQ: So the kind of method I described earlier, the 2011, 2012 one, effectively all provides are constraints. It doesn't completely solve the problem. All it tells you is are two amino acids in proximity to one another? It's not even perfect. So sometimes the predictions are actually incorrect. But it provides sort of high-level constraints.

What typically was done in 2012 and 2011 was to take these kind of constraints and then feed them through the conventional pipelines I was describing, these three million lines of code pipelines, which then uses additional information and physical simulations and so on to try to fill the protein. So the paradigm was not sort of fundamentally altered in terms of how you simulate and so on. You just provided more information to help this process along.

Where deep learning helps is to basically refine these prediction, so instead of – Like I said earlier, these predictions are just kind very coarse-grained. They're two residues are in proximity to one another and they may or may not be accurate. What deep learning at least up to, say, 2018 had done is basically take these kind of raw predictions or raw calculations rather from the coevolution set and then turn them into more accurate predictions of context.

[00:45:34] JM: So tell me if I understand it correctly. With the coevolution understanding, you basically get to an inflate the amount of data that you have available for these deep learning techniques, which we know by now are unreasonably effective on large datasets.

[00:45:53] MQ: Well, it's actually interesting. It hasn't really been used that way that you just proposed. That is not a bad idea in a way. One could use this information as outputs instead of say, "Okay. We're going to predict these things." But that's not typically how they are used. They are rather used as inputs. So they don't really increase the number of data points, because we're still with those 50,000 or what have you. But they add additional information per data points. So they make your input modality richer, if that makes sense. So instead of just having sequences, you also have these evolutionary contexts.

[00:46:23] JM: Okay. Not to bury the lead here, but you had this really detailed blog post that was a retrospect on CASP 13. CASP 13 was one of these computations, these protein structure prediction competitions. What was unusual about this competition is that it was by won by DeepMind, and DeepMind is the company that was acquired by Google and they're an innovator in deep learning techniques. They were able to create the first system that was able to beat every human, or the best humans at Go, for example. That was Alpha Go. They later innovated upon that with Alpha Zero. Then they, at CASP 13, managed to decimate the competition with something called Alpha Fold.

Explain why this is a significant event.

[00:47:22] MQ: Sure. I would it's maybe significant in two ways. One is the fact that they did well. I don't think – The improvement is probably in the order – I mentioned this in the blog, on the order of like 2X. If you look at historically how much CASP has been kind of improving from year to year, this sort of doubled the expected rate of improvement. The gap they had between them and the second closest competitor was much higher than usual. They certainly didn't solve the problem, but they made substantial progress. For that, it was significant.

The other reason maybe perhaps was kind of sociological, is that it's a company. This field has been dominated by academics for its entirety. It's not, to my knowledge, been any prior entrant from industry. So this was like the first time that a serious kind of industrial lab decides to enter the space.

[00:48:09] JM: In computer science, we've been watching – How would you say? I guess the evolution of the dynamics between academia and industry for a while. Now, I mean, you look at like – I think, the MapReduce paper comes to mind, the Google MapReduce paper, this thing that came out of Google that totally turned computer science on its head in some ways. I think probably there's some fear or envy or – I don't want to use those words, because that makes it sound like a negative event or adoration perhaps of the fact that such a beautiful piece of research could come out of a corporation. Since then, there's been this interesting dynamic between academia and industry.

How do you see the dynamics between academia and industry and, I guess, historically over the last 10, 20 years, and how does the deep mind win of CASP change your belief in those dynamics?

[00:49:05] MQ: Yeah. Again, there's a lot to unpack here.

[00:49:07] JM: Sorry about that.

[00:49:08] MQ: No. No. No. It's a great question. I mean, it's interesting you mentioned computer science, I'd like to think of myself as a computer scientist as well. I've been following

how it definitely transformed particularly in machine learning in the past four years, many people have been maybe slightly upset that, “Oh, all these academics are leaving in some ways academia and going to industry.” Arguably, now the best machine learning laboratories are Google and Facebook and so on and not Stanford and MIT and so on. So definitely, there’s been that shift. I think biology maybe is lagging a little bit a bit in a few years. So I think potentially we might see something similar, maybe.

What I would say in biology sort of historically is there’s definitely been a lot of interaction between industry and academia particularly on the translational end, in sort of biomedicine. Because pharmaceuticals obviously have been very much interested in developing, needless to say, drugs. So far as that intersects with research, it’s often the case that there’s been kind of synergistic relationship.

What’s unusual about DeepMind and sort of just kind of what the historical context, is that there’s not been that much – There’s not really been pretty much any involvement from industry, especially big companies, in basic, basic research, biological research. They haven’t [inaudible 00:50:20] companies that have done, I would say, sort of fairly basic biological research, like Schrodinger for example, but they’re smaller. These kind of large companies have not really been kind of interested in this space. In that sense, it’s sort of a new dynamic that we see interest in sort of fairly basic problems in biology.

[00:50:37] JM: Tell me more about your personal reaction to the DeepMind win, like in terms of both emotion and how you’re thinking about your own career. How did you feel about that?

[00:50:48] MQ: Yeah, yeah, sure. It’s a multi-stage process in a way. So I was also competing in CASP and I’ve been sort of developing kind of deep learning approaches as well. So I have sort of skin in the game in a way. So from that perspective, obviously, I was sort of disappointed in a way, you could say. But that maybe will have more to do with me not winning as supposed to sort of them winning, if that makes sense. It’s just because we’re all competing for the same thing. So that was that initial disappointment.

I mean, on the whole obviously, this is a great news for the field, both because they're bringing more attention to it and because they've helped move it forward. In that sense, at the end of the day we're all in this to see protein structure get better. I think that's in that win.

Longer term. I mean, I think for the time being, I'm not going to sort of alter my research reaction very much. I mean, I think it would be silly to see how engaged DeepMind and everyone else in this space. I think what we will see I suspect, and myself I think I'm going to listen to my own advice here, is that just like in computer science where there's been maybe some bifurcation in the sense that the kind of very compute-heavy approaches that just require a ton of data and just tons of resources to see through, those have become the domain of Google and Facebook and so on.

I think in academia, we can see the same thing, where people are going to move away from this sort of resource-heavy approaches and shift toward things that I may be a bit more kind of logical, a bit more novelty and sort of creativity-driven. Maybe at the end of the day, don't work as well, say, as kind of the resource-heavy approaches. But they point the path forward kind of toward more innovative directions that would be useful down the line. So that's my projection for how this will play out.

[00:52:28] JM: That's a fascinating analysis in light of some of the trends that I've been seeing in new software companies, where there's certainly an aversion to creating a company or a module or a software project that cannot compete with the data advantage and the resource advantage of these giant companies. In some sense, it's useful on the industry as a constraint, because it forces people to think a little bit more creatively about how to build businesses in light of these enormous monoliths.

We're running up against time. I'd love to know a little bit more about what you consider, I guess, the ongoing evolution between this academic and industry world. Specifically, what are the durable competitive advantages? If we think about academia versus industry, areas that are competing with each other, what are the durable competitive advantages of academia?

[00:53:22] MQ: I think the primary one is the fact that you – I mean, how true this is is not entirely clear. But at least in principle anyway, let's just say. In academia, you sort of have

freedom to follow your heart instead of try to pursue any kind of approach that appeals to you. Well, I think in industry, I think there is a more of a pressure to sort of produce results that have a wild factor. So I think in that regard, academia may have an advantage in being able to sort of pursue these sort of more out of the box or sort of fairly kind of unorthodox approaches.

To be clear, and I think this is what makes this era I think particularly interesting, is that you do have companies, even maybe you call them even institutions, so the right word, like DeepMind, which are actually pretty interested in doing basic science. That's what I think is quite different about this and this kind of harks back to the days of bubble ads and so on. I mean, they're quite different in a way than, say, Google research. Kind of the historical construed, because as far as I can tell, they're not under any pressure to sort of immediately product a product.

That I think is they do provide a very compelling competitor to academia in that regard. Maybe the last thing I'll add on this point is where I think they might enter some trouble if they try to kind of go beyond these very well-defined problems. So the nice thing with protein structure prediction is that the question has already been figured out for you. We know what the question is. We just need to solve it. That's quite rare actually in science. Most of the time, it's asking the right question that's the hard part.

I think if they decide to sort of move in the direction where they're trying to kind of effectively come up with their questions instead of broaden the breadth of the questions they try to tackle or the problems they try to tackle, that's where I think they will get into some trouble. But so long as very well-defined, I think they're going to provide some very tough competition.

[00:55:17] JM: Now, I'm really hoping to eventually do a show with someone from DeepMind and maybe in that episode we can go a little bit deeper into Alpha Fold and how it works, because I think with you and I, we haven't really gone into the depths of how that works. At this point, it's probably beyond the scope of this conversation. So maybe we could just kind of close on some thoughts about that fact. I've tried to do shows with DeepMind, and it's been pretty tough. That's totally their prerogative. That's absolutely fine. But I would say it does kind of illustrate one of the durable competitive advantages of academia, which is its openness and it's kind of lack of the pressures of industry. You see the gradient between these two worlds I think being explored by OpenAI. I think open AI is kind of the first – I'm brutally happy that OpenAI

exists. I didn't really realize how important it would be when it was started, I guess, four or five years ago. But OpenAI, you kind of have this default open environment in contrast to this sort of impenetrable opacity of DeepMind that only gains shades of transparency through promotional blog posts. So I guess as a closing topic, how do you see I guess the rising competition, if you would describe it that way, or coopetition between open AI and DeepMind.

[00:56:48] MQ: Yeah. Well, I mean, it's interesting you say all these, because in some ways I think the difference is maybe going to shrink in a bit unfortunately, because OpanAI just recently decided they're going to become a limited for profit, right? They have this mechanism where they –

[00:57:00] JM: But for fairly virtuous reasons.

[00:57:02] MQ: Quite possibly. But, I mean, the problem obviously is structurally once things change and they sort of depend on the goodwill of the people involved, and obviously that changes overtime. So it's hard to go and keep.

I mean, it's an interesting point. To be honest, I think we suffer from this in academia a little bit. I mean, I talk about this in the blog post, where I think some of the things that held back protein structure prediction work in academia is the fact that people are a bit secretive and they don't sort of polish their results until every two years or whatever when this CASP thing happened, and I think that held us back. I think computer science has been much healthier in a way, because it's been much more iterative and people publish much more quickly.

So I would say as an ecosystem – Initially when I was answering your question, I was thinking from the perspective of an individual lab. But as an ecosystem, I do think academia does provide a very compelling alternative, because you do have – Like you just said. It is much more open. Of course, as big as DeepMind maybe, it's not going to sort of compete with this bizarre – Bazar, rather, of different ideas to kind of take an open source analogy. I think [inaudible 00:58:05] to how academic research is done [inaudible 00:58:10] new ideas bubble up all the time, and I think in the long run will I think improve to be durable.

[00:58:18] JM: Mohammed AlQuraishi, thank you so much for coming on the show. It's been enlightening to talk with you.

[00:58:22] MQ: Oh, thank you very much or having me.

[END OF INTERVIEW]

[00:58:27] JM: This podcast is brought to you by wix.com. Build your website quickly with Wix. Wix code unites design features with advanced code capabilities, so you can build data-driven websites and professional web apps very quickly. You can store and manage unlimited data, you can create hundreds of dynamic pages, you can add repeating layouts, make custom forms, call external APIs and take full control of your sites functionality using Wix Code APIs and your own JavaScript. You don't need HTML or CSS.

With Wix codes, built-in database and IDE, you've got one click deployment that instantly updates all the content on your site and everything is SEO friendly. What about security and hosting and maintenance? Wix has you covered, so you can spend more time focusing on yourself and your clients.

If you're not a developer, it's not a problem. There's plenty that you can do without writing a lot of code, although of course if you are a developer, then you can do much more. You can explore all the resources on the Wix Code's site to learn more about web development wherever you are in your developer career. You can discover video tutorials, articles, code snippets, API references and a lively forum where you can get advanced tips from Wix Code experts.

Check it out for yourself at wicks.com/sed. That's wix.com/sed. You can get 10% off your premium plan while developing a website quickly for the web. To get that 10% off the premium plan and support Software Engineering Daily, go to wix.com/sed and see what you can do with Wix Code today.

[END]