EPISODE 1267

[INTRODUCTION]

[00:00:00] JM: Arun Kumar is an assistant professor in the Department of Computer Science and Engineering at the University of California at San Diego. His primary research interests are in data management and systems for machine learning and artificial intelligence-based data analytics. Systems and ideas based on his research have been released as part of the Apache MADlib open source library, shipped as part of products from Cloudera, IBM, Oracle and Pivotal and used internally by Facebook, Google, LogicBlocks, Microsoft and other companies.

Arun did his undergrad in Computer Science and Engineering at the Indian Institute of Technology Madras, and then his MS and PhD in computer science at the University of Wisconsin Madison, where his thesis research explores problems at the intersection of data management and machine learning with a focus on problems related to usability, developability, performance and scalability. In this episode, he joins us to discuss data management systems and artificial intelligence.

A few announcements before we get started. One, if you like Clubhouse, subscribe to the Club for Software Daily on Clubhouse. It's just Software Daily. And we'll be doing some interesting Clubhouse sessions within the next few weeks. And two, if you're looking for a job, we are hiring a variety of roles. We're looking for a social media manager. We're looking for a graphic designer. And we're looking for writers. If you are interested in contributing content to Software Engineering Daily, or even if you're a podcaster, and you're curious about how to get involved, we are looking for people with interesting backgrounds who can contribute to Software Engineering Daily. Again, mostly we're looking for social media help and design help. But if you're a writer or a podcaster, we'd also love to hear from you. You can send me an email with your resume, jeff@softwareengineeringdaily.com. That's jeff@softwareengineeringdaily.com.

[INTERVIEW]

[00:01:55] JM: Arun, welcome to the show.

[00:01:57] AK: Thank you for having me again, Jeff. Nice to be back.

[00:02:01] JM: It's great to have you. You run a lab at UCSD. It's called the ADALab, Arun's Data Analytics Lab. Tell me about what you focus on in your lab.

[00:02:13] AK: Sure, so the data lab, also called the Advanced Data Analytics Lab, focus on basically improving machine learning and artificial intelligence-based data analytics from a holistic standpoint of concerns such as scalability, usability, ease of deployment, ease of development, and so on. And I call all of this democratization of machine learning-based data analytics. I look at it from the vantage point of data management, databases, data systems, and I struggle the whole system stack aiming to improve two specific things, system efficiency, by which I mean, the resource needs, the resource costs, the time runtime requirements of all the tools involved, and also human efficiency. And by which I mean improving the productivity of the people involved in the process, the data scientist, the machine learning engineers, the data analysts, the domain scientists, and so on. And so that's what my lab focuses on. Building software systems, building abstractions, building tools, devising new algorithms, and also studying the theory behind things the empiricism, the experimental analysis behind this whole process of making machine learning-based analytics easier, and faster, and cheaper for people.

[00:03:30] JM: How do the problems that you study in the academic domain compared to those problems found in industry? Are they the same?

[00:03:43] AK: Right. So there is a close synergy between practice and research here. And the reason the synergy exist, and I'm telling this based on my conversations with over three dozen practitioners in this context, right? So enterprise, health, web companies, cloud companies, database companies, domain scientists, and others, digital humanists. Based on these conversations, what I find is, the science of this whole space is still lagging. It's still not where we wanted to be. And the analogy I would give is relational database systems circa early 1980s, the science of relational database systems was still lagging and people were building database systems. And then relational model came along. And then people started building SQL and relational operators and parallel database systems and all of that. And look at how far we have come in the last 40 years with relational database systems.

Machine learning systems and the end-to-end lifecycle of machine learning applications, it's kind of in that context, where we understand the theory and the algorithmics of how machine learning algorithms work. But the stuff that goes around it, all the way from what I call the three stages of the ML lifecycle, sourcing, building and deploying. The sourcing stage is how to convert raw data in your data lakes, data warehouses, databases into an ML-ready data. And then how to convert that into a prediction pipeline? That's the build stage. How to integrate a prediction pipeline with your application and monitor and oversee the life cycle as the data and application evolve? That's the deploy stage. These three stages, I think the amount of scientific inquiry we have is still not there at the level that we want it to be.

Now, practitioners, obviously, they have to get their job done until they get about doing these sorts of things already. However, the principles behind them, whether what they're doing is accurate enough, whether what you're doing is, in the sense, can we reduce the cost? Can we improve their productivity? Those sorts of questions, a lot of computing research is fundamentally about those sorts of questions. And I think in the ML space, that is only starting to take off. And so in that sense, I see a lot of my work as actually laying the foundations of the principles and the systems that are needed for this holistic picture of ML. And that cannot come about in a vacuum. You have to interact with practitioners to understand where the bottlenecks are. You have to interact with practitioners to complete the loop after you come up with new techniques and solutions to see that they actually work in practice, and what you can learn from it, and how you can improve research and the principles in the next iteration. And so this synergy between research and practice that really excites me about the space. It's not kind of some arcane academic stuff that you sit in a corner and do and 20 years later somebody uses it. Here, you're actually coming up with very fundamental research ideas that have long-term potential, but also immediate applicability in practice. I hope that helps understand the space better.

[00:06:43] JM: That makes sense. So let's start with the – In the lifecycle of data, Advanced Data Analytics, can you describe, the sourcing, building and deployment. Let's start with the sourcing. What are the problems that exist in the data sourcing step of machine learning?

[00:07:01] AK: Okay, sure, I'm happy to chat about this. Also, for folks who are interested, I also have a graduate course on data systems on machine learning, CSE 234. And the lectures and

videos are all up on my course webpage. It's on YouTube. So feel free to go check. Take a look at it. I walk through these stages in detail for my students, and I will just recap what I teach there.

For the sourcing stage, from my conversations with the practitioners, what I've learned is there's a huge set of disparate activities. A lot of them revolve around data management issues. And some of the most prominent and common activities include procuring the data, procuring the data in the sense you want to – Or acquiring the data, acquisition of data, going through the hoops of access control, and system administration, and all of that authentication, just to get the data you want.

It's shocking to me that in many enterprises and web companies, people can't even see the data they want to see, because different teams are on different data. So even figuring out which data is valuable is the first bottleneck. You manage. It might ask you to build a model for some particular prediction task. But do you even have the data you want to build that model effectively? And so do you know where the data is in your organization to get that sort of model built? So that sort of acquisition itself is the first challenge. There are a lot of research questions there in terms of data search systems, metadata management, data discovery, schema management, all of these things. So that's one.

The second one is after you acquire your data that you want, you need to be able to transform them. And that's where data preparation comes in. Data Preparation is basically reformatting the data. A lot of mundane things, figuring out the features, stitching the features together into a file format that's useful for your ML tool. Again, a lot of database style stuff happens there. And machine learning algorithmists, they look at all this as grunt work. They bundle all of this under this giant moniker called data cleaning. But actually there're different steps that take place. It's not all just data cleaning. So there's reorganization of data, there's reformatting of data. All of this comes under data organization or data preparation. Then there's actual cleaning, which in a statistical sense means fixing issues in the data so that the bias variance tradeoff doesn't get affected negatively, imputation of missing values, inconsistency in the values, normalizing the values of features like, say, California versus CA. These sorts of stuff that statistics and database people have been studying for decades now. That is inevitable, especially when you're dealing with tabular data. So that's one.

The other major task and the sourcing stage is sometimes optional, because many times the data distribution provides you this, is the task of labeling. In many enterprise settings, you don't have label data available. You have to sit and label them manually. Increasingly, there's a lot of research on programmatic labeling, crowdsourcing, weak supervision, semi kind of distance supervision, active learning. All of these things are now becoming more common. So that's another part of sourcing. Getting the label in place.

And then once all of that is done, you have to kind of – The data is now ML-ready. You're still not done. You still have to engineer features in many contexts. And for tabular data, that involves very standard mundane things, like one hot encoding, similarity encoding, bending, and all of that. But for other unstructured data sources, you have to do more sophisticated feature engineering, like embeddings, or applying transformers in different ways, extracting CNN features in different ways for images, and fusing these features together. And then that's where the boundary is between the sourcing stage and the building stage, because a lot of the feature engineering decisions are no longer about just sourcing your features, but now representing your features for the model to learn better. So I hope that gives you a bird's eye picture of the major bottlenecks and the major technical tasks in the sourcing stage.

[00:10:59] JM: Can you describe some of the main tools used in the sourcing stage of data engineering?

[00:11:05] AK: Yeah, sure. So in terms of tools, for data acquisition, I think there's a dearth of tools. There's not that many that people use that are standardized. There are catalog management tools. There are metadata management tools that a lot of people use. MLflow is, for example, one of them. And that has seen a lot of traction recently. There's TensorFlow Extended as well. And so identifying and registering models and datasets in your organization is pretty useful. And so these are two common open source tools that I've seen being used for data acquisition, for metadata management and so on.

In the preparation stage, reorganization stage, there is, again, not that many commonly used reusable tooling infrastructure. Some of my own research actually studies, how do we automate some of this from a principle standpoint, but not just automation? How do we measure that

automation is good enough? So how do we benchmark it? So there's still a lot of active research going on in this space. As far as I can tell, a lot of the tooling in the preparation space is all human in the loop and self-service-oriented. Like Trifecta is one example. Tableau data prep is another. So these are some common tools that I've seen people use.

In the cleaning context, a lot of this is very manual. Increasingly, there's some tooling for semi-automation of some of the cleaning tasks. Amazon has this DQ tool that came out in the last couple of years that has seen some traction. Databricks has also shipped this. There are some machine learning based cleaning solutions that are now coming around. All of these things put together. Some companies have started to automate them in some ad hoc ways. The workflow management tools, and a lot of people like to hate on Airflow, but Airflow is pretty common. And people like to automate some of these workflows using those sorts of tools. Kubeflow is also being used for data preparation and data cleaning.

I see a lot of tech companies building solutions to automate them using role-based heuristics. Salesforce, for example, has Transmogrify and Einstein for the data prep and feature engineering and stuff like that. Amazon came out with their AutoGluon. They also have their Autopilot. Google has their Cloud AutoML and AutoML Tables. So there is increasing infrastructure being built around some of these tasks. But a lot of the work is still pretty manual. I mean, if you look at surveys of data scientists, ML engineers, and so on, a lot of them say they spend a huge amount of time on data-related grunt work and data-related issues. And this is partly why. It's very hard to automate the whole thing very robustly and accurately. And so the tooling and the infrastructure is only starting to – You're only seeing the assembly of the sort of tools starting to grow now.

[00:13:57] JM: As an engineer goes from the data sourcing phase to the model building phase, what is their day-to-day work look like? Is it mostly programming? Or is there a lot of observation and analysis of the data? Tell me about the building stage of the lifecycle of data analytics.

[00:14:21] AK: Yeah, sure. So the interesting bit for a lot of data scientists and ML engineers is the building stage and the deployment analysis, postdoc analysis. Now the building stage is where a lot of machine learning and data mining class is focused their efforts — things like how

to prepare features. How to do feature selection? How to do algorithm selection and algorithm search? How to do hyper parameter tuning? These are the things that go into the building stage.

Now, fundamentally, from a learning theoretic standpoint, the purpose of the building stage is fairly simple. You have your ML-ready dataset pairs of inputs and outputs or maybe unsupervised setting to self-supervised. And then the output is a desired prediction function that satisfy certain metrics. Usually the most common metric is accuracy, prediction accuracy. It could be various forms of a diagonal or **[inaudible 00:15:13]** precision recall and all that. These days, we also have to worry about fairness metrics and disparate impact. Inference latency is also an important metric. Throughput of training is also a metric. So there's so many metrics these days.

So data scientists, in the building stage, the first thing they need to worry about is what is their application goal? And it's not just a routine matter of taking their ML knowledge from the classes and then say, “Okay, I'm going to build a new deep neural network.” That is not their application goal. The application goal is once you have built a prediction functional pipeline, how is it going to achieve something for your application? Understanding that part, the objectives itself is a big bottleneck, for a lot of users of machine learning.

Often, there's a lot of conflict in terms of what the application goals are. And so figuring that out and then working backwards from that is a big part of the building stage. And often, that involves a lot of search and a lot of comparisons, empiricism, a lot of explorations. And I call this whole phase the model selection phase. And so the model selection phase is ultimately what you do to devise different hypotheses and obtain the prediction functions that you want. And this hypothesis search results tuning a lot of things, everything that you tune. The way you impute the data, the way you bin the data. The logistic versus neural networks SS3 that you select, the hyper parameters that you tune, every single one of them affects the bias-variance noise decomposition.

And ultimately, what a lot of this building stage is about is tuning the knobs up and down on the bias, on the variants, on the noise to ensure that your application metrics are satisfactory. And so the way I look at as this whole phase of building involves applying the knowledge that people learn both the algorithmics and the theory of machine learning and connecting them with the

application goals that they have, which often you learn only in the real world. I mean, how much can machine learning classes teach about thousands of applications of machine learning, right? And then connecting that, bridging that gap via tools, via mechanisms, that allow you to meet those application metrics in an effective way. And often, that involves huge resource expenses. So especially in the deep learning context, often I hear people don't even want to tune hyper parameters or algorithms or **[inaudible 00:17:31]**, because it's just too expensive for them. And so resource costs need to come down. Business expenses need to come down. The amount of time they spend on model building needs to come down.

Thankfully, a lot of feature engineering in the unstructured data context has been automated with deep learning. So a lot of that has gone away in the NLP world, in the vision world. But in the tabular data world, that's still the case. You have to do it a little bit more manually, because of interpretability, and explanation and all these other concerns. So during the model building phase, there is always a Pareto Front. Data scientists have to navigate that Pareto Front. It's not just accuracy. You have to worry about all these other concerns, like I said, fairness, inference latency, training cost, interpretability, explainability. All of these things matter. And there is some increasing tooling to help navigate this Pareto surface. There is also tooling to help debug model accuracy and build better models. So there is a lot of action even in the building phase for this overarching application that I'm talking about, application lifecycle, beyond just the basics of machine learning algorithms, beyond just the theory of machine learning itself. And so a lot of that is happening in the ML systems world, in the database world, in the ATI world. And this is also part of this new principle that I'm talking about for the whole field.

[00:18:53] JM: I want to get into some of the projects that you've worked on specifically at the ADAlab. But since we've touched on sourcing and building, let's now evaluate the deployment step somewhat. Tell me about the deployment step, the frictions of the deployment step, and the typical infrastructure that data analytics models are being deployed to.

[00:19:19] AK: Okay, sure, happy to. I mean, this is probably the step that I'm least familiar with, because I'm in academia, but I keep in touch with all my friends in industry who work on this kind of stuff. And shout out to the folks who have helped me understand the deployed state most, Vartak, who has the startup Verta. **[inaudible 00:19:36]**, who was at Google, now at Databricks, Mate at Databricks. And recently Shreya Shanker, who was at Google. So I learned

a lot from these folks on model deployment. One thing to note is the scenario for model deployment has been around for decades. It's not a new thing. This whole notion of ML ops. It is being relatively simple in the ML avatars in the past, because, typically, like you build a model with SaaS or something like that, and then you integrate it with your relational DBMS, or your data warehouse. That's how it has been going on since the 90s. That world was much simpler for model deployment. You just take your model that you've learned. You do inference on the database system or on the data warehouse system for deployment.

But over the last couple of decades, and this is something I explained in my course as well, the complexity of model deployment landscape has exploded. And the reason it has exploded is because of cloud computing, because of the web, because of the Internet of Things, and heterogeneity of hardware, heterogeneity of models being used. All of these things have caused an explosion of complexity. And especially in the web companies, app companies, the IoT world, deployment is very heterogeneous. I mean, any software deployment is very heterogeneous, and ML is no different. It is also a form of software. So if you want to do ML-based prediction for certain things, that is a form of software. And so all the complexity of software deployment, amplified, manifests itself in the ML ops world.

And the reason it's amplified is it's no longer deterministic. It's probabilistic. And you will have to log the predictions and the bucket in the future. You will have to worry about thresholds on probabilities after the model has made its prediction. And the thresholds may not always be obvious. So there's added complexity on top of deterministic software for deployment of ML software. So what I see happening is industry is trying to standardize a lot of this in the form of these tools like MLflow, or TFX, TF Serving, to make it easier for people who do the model building and the data sourcing to not worry about it too much. So basically, they build a prediction pipelines. And then they have a standardized environment or infrastructure where they hand it off, and it gets into AWS where they call it model endpoints.

So these sorts of endpoints for connecting the deployment stage, versus the largely offline phase of sourcing and building where you set up the infrastructure is starting to happen. And that's a good thing. I think the tooling is still evolving. It's still not settled yet. It's still an active space. There’re a lot of startups that I see in the ML deployment landscape. And the cloud companies are also going after this They are trying to standardize this. It's still going to be a

heterogeneous mix. I don't think it's going to be a one size fits all. And the old world of deploying models into your data infrastructure is still important. A lot of enterprises, their data is under data warehouses and database systems. And they still want to integrate their models into that infrastructure. A lot of web companies, of course, have their cloud-based solutions. And so they wanted to deploy it there.

All of this, increasingly, there is also a push towards more rapid evolution of models. And the more rapid evolution of models means, as the data distribution evolves, as the concept drifts, you want to make sure that the life cycle is kind of kept fresh. In the past, this used to happen every quarter, every month, if you're lucky. But now they want to do it every day, or even every hour, or even minutes in some cases. And so now your deployment infrastructure becomes even more critical, because you want to do very close to real-time kind of updates to your data, updates to your model. And that requires refactoring your deployment infrastructure in a way that is amenable to it. And so, increasingly, I've seen these ML platforms and feature stores worlds evolve in that direction where they're trying to create something that is more reusable, that is easier to maintain, that is easier to scale and study separately. So this separation of concerns is starting to happen.

And again, I think as an academic, I only get to see so much of the deployment landscape, because I don't have, I don't know, hundreds of millions of people accessing manual services that these giant tech companies have. And I only know so much about the deployment side. So a lot of my research has focused on deployment of models in the data systems context. But I haven't really studied that much about the online prediction serving or those sorts of settings.

[00:24:09] JM: So now that we've evaluated the high-level perspective of the data analytics processes that your lab focuses on, we can touch on some of the specific projects, one of which is Project Cerebro. Project Cerebro is focused on making artificial neural network model selection easier to work with. Can you tell me about the problems with artificial neural network development and what you're building with Cerebro?

[00:24:43] AK: Oh, absolutely. Now, to talk about Cerebro. So, Project Cerebro is the one that was funded by my NSF Career award. And I was just surprised at the level of interest that I've seen in various contexts, not just in the basic science world, but also in the public health world,

because we have an NIH grant that also helping funded in the industry. VMware gave me a couple of gifts to help fund that work. And so I've seen a lot of interest from various sources. And the reason for this is fundamental. The way artificial neural nets are built is highly empirical. And, I mean, most of machine learning is pretty empirical. Model selection stuff is pretty empirical, even for things like kernel machines and all of these things that have a lot of theory behind them.

The reason it is so empirical is the bias variance-noise decomposition that I talked about earlier. The hypothesis space of prediction functions that you deal with is directly controlled by the neural architecture. And so the larger the model, the more neurons, the deeper it is, the richer the hypothesis space. And so you navigate this bias-variance decomposition by tuning the hypothesis space.

On top of that, the hyper parameters that you need to tune for optimization, stochastic gradient descent is the optimizer of choice. And there are various flavors of it **[inaudible 00:26:02]** whatnot. Every one of them has hyper parameters. And these hyper parameters control how well your model learns. It controls several other things. It controls the variance of the bias-variance noise decomposition as well.

And then on top of that, you have feature representation. Like, say, if you're doing time series analytics, and this is something we've been doing a lot with our public health collaborators here at UC San Diego, even have a couple of public health Journal papers that we applied Cerebro, and we saw some interesting lessons. Say, windowing of the features, that is a feature representation decision. That also affects the bias-variance noise decomposition. If you aggregate too much, you probably introduce more base noise.

And so these sorts – Tuning these three things that affect your outcomes, affect you prediction accuracy, is inherently empirical. You cannot tell upfront with 100% confidence that if I do X, it will always be better. That's very, very rare that you can say this. Like even if you want to say I have this training data set, I want to build the deepest model possible. You go and build the deepest model. Okay, so these days, there's this double descent phenomenon. You can actually overfit to the training data entirely and still generalize well. It's not easy to reach the double descent phenomenon. Sometimes you don't end up getting into that regime. And even if you do

get into that regime, if you have a giant bloated model, your DevOps people and your ML ops people are going to be angry with you. They're like, “Why are you using such a giant bloated model for my inference task, which I have 10 millisecond latency?” or something like that.

So just the Pareto surface comes back to bite you. There are multiple conflicting criteria that a scientist has to satisfy. And so eventually, they will have to inevitably explore these alternatives for all sorts of things. Now, to be more concrete, in our public health use case where we were building – So the data has accelerometer time series. And they have like almost a terabyte of this. They have labeled this data. The prediction task is identifying whether the person is sitting or standing and moving based on the accelerometers that they wear on their hip. And the cohorts that they are studying, and these are important cohorts, or like people who are survivors of breast cancer, people who have had obesity, people in assisted living facilities. In many of the cohorts that they're looking at right now, the goal, the public health goal is to understand how much exercise are they getting throughout the day? You don't want to do this in a very personal intrusive way by monitoring them like as in-person. And so these accelerometers are helping public health scientists understand their behavior throughout the day in a postdoc fashion.

And so they have this label data of time series. The output is also a time series. And previously, they had a random forest baseline based on heavy duty handcrafted time temporal features. And for a binarized task, are they sitting versus non-sitting? The random forest I think gave an accuracy in the high 70s. But we came in and we did some model building for applying CNN LSDM hybrids. So convolution neural nets, plus long short term memory nets. And these are the state of the art, among the state of the art for time series prediction tasks. And we reached 92% accuracy. And that's a lift of almost 14%, 15%, or a binary test that's rare. And they were really surprised. We were really surprised, and they were very happy about it. And they have been applying this to many cohorts now. This accuracy is the state of the art for this task in their field. That's why these journals have been excited about this as well.

And the way we managed to do that is with model selection. And we realized that the model selection was the key bottleneck in raising this accuracy here, because the architecture of the CNN LSDM, the hyper parameters, the windowing, all of this defining. The correct kind of exploration process itself was integral. We didn't have to do handcrafted-feature engineering. We had to do handcrafted architecture engineering, and that is inevitable in deep learning.

On top of that, coming back to the metrics point that I made, you would think, “Okay, binary classification loss is the only accuracy to optimize for.” You would be wrong. These public health scientists have a dozen metrics that they optimize for. These dozen metrics include things like precision, recall, transition, precision, transition recall, some other aggregate metrics from the public health standpoint. And when you do this model selection sweep over different architectures and hyper parameters, you would realize, you will end up with Pareto optima. Some models work better for some metrics, but the other models work for other metrics.

And eventually what we ended up doing was analyzing all of this and then building an ensemble of the predictions for a subset of the models. And so this sort of tradeoffs that are inherent in a real-world application of ML, that is sort of what we realized in the Cerebro Project. We want to make this sort of exploration project, exploration process, high-throughput. We want to reduce resource costs. We want to reduce runtimes. And the implication is pretty clear. If you reduce runtimes, if you reduce resource costs, people will be more willing to do more of the sort of exploration and reach better accuracy and better metrics. And so that's what we saw in this project. That's the direct implication. That is what motivated us to build out the Cerebro system.

And then I realized that this philosophy is actually more powerful than just in this context. This philosophy is fundamentally building a set of models at a time, rather than just one model at a time. And going back to the bias-variance noise decomposition, it's fundamental in learning that you cannot escape it. It happens over and over in many contexts. One other context that happens is transfer learning. Let's say if you want to use a pre-trained CNN to get features for an image. Which layer of the pre-trained CNN are you going to explore for your feature representation? That is a decision for your bias-variance tradeoff. And so that requires exploration. That is a model selection step.

Another thing I've seen as data scientists like to build separate models for separate populations. Like say, one model for California, one model for the East Coast, or whatever, or different countries. This is what I call learning over groups. Instead of taking the whole population building one model, you build separate models for separate subpopulations. That is also a bias variance tradeoff. The individual subpopulations distribution could be maybe a little bit simpler than the whole population. So you end up with a better base-variance tradeoff.

And so all of these sorts of scenarios that I've seen, where you tune these knobs for controlling the hypotheses base, controlling the bias-variance noise decomposition, call that model selection management. And so the Cerebro project was really aimed at making this model selection management process easier. Making it faster, reducing resource costs, using database style ideas. And that's the vision paper we wrote at the CIDR, the CIDR Conference earlier this year, where I laid out this layer design of the Cerebral system. The what of the model selection process is specified at the level that which at which data scientists are comfortable with using the tools that they are comfortable with. So they don't have to change their usage of TensorFlow or PyTorch, or Kerris. But under the hood, we rewrite the computations in a way that improves resource efficiency, reduces runtimes, all of these good things that we know from the query optimization land. And then even below that, you can now execute it on various environments. You could run it on your in-house cluster. You could run it on the cloud. You could run it on database systems, or Spark, stuff like that. This sort of layer decouple design with query optimization at the heart is what we pursued in the Cerebro system. And it all came back to that original thing that I mentioned, which is model selection is inevitable because of the bias-variance noise decomposition being extremely hard to predict upfront. So empiricism and exploration is essential in the artificial neural network world.

[00:33:42] JM: Can you give an example of like an application that would be ripe for using Cerebro, for using the tool to improve model selection?

[00:33:54] AK: Yeah, sure. So the public health application was one where they had this time series data, and they wanted to predict the activity. And now you could go ahead and define your architectures and your hyper parameters in the Kerris API's of Cerebro. And you could just run this. And the code is now released. We even have released a documentation and examples on our website. Feel free to go check it out. And this can be used in many other domain science settings. We are also collaborating with material scientists where they want to analyze simulation data. And so now your input would be simulation logs, and your output would be some physics-based phenomenon. And so now you could specify that as input and then define different neural architectures and hyper parameters that you want to try over and then give that to Cerebro. And it will execute this at scale for you.

Other application scenarios that we are looking at right now in collaboration with our colleagues at VMware, they actually adopted the central ideas of Cerebro in the Apache MADlib project. And this is in RDBMS ML tool that Pivotal has. And they have been talking to different enterprises for natural language processing and computer vision use cases for enterprise datasets. A lot of these enterprise data sets, they have structured data. They also have text data. They also have some images in some contexts. And it's stored on data lakes. It's stored on data warehouses and database systems. And there, they want to do very simple things like sentiment mining, or named entity recognition. These are machine learning tasks. And so they could just specify the off-the-shelf transformer architectures, doing hyper parameters for fine tuning on the data sets, and then give that to the Cerebro API's, and then they could build it at a higher throughput.

So these are sort of some concrete applications that we have been studying or exploring and working on ourselves or with our collaborators. But really, I mean, it could be applied to virtually any prediction task. Any prediction task where you would need to build a model and need to tune hyperparameters, need to tune the architecture, Cerebro is useful there. It's really a very general system for all these sorts of model selection tasks, where you're building a neural computational graph. And the neural computational graph frameworks that we currently support are TensorFlow and PyTorch. It could easily be extended to other things in the future. But currently, we are focusing on TensorFlow and PyTorch. They are the most commonly used frameworks for specifying neural computation graphs, and executing them and training them with SGD. And so we are reusing the infrastructure of TensorFlow and PyTorch for the SGD implementation, for the neural computational graph compilation and execution. And what Cerebro really orchestrates is the scalability part of it, is the data handling part of it, is the distribution part of it.

[00:36:40] JM: In working on Cerebro, did you engage with any large corporations about how the problems that you were tackling with it compared to what they were encountering?

[00:36:53] AK: Actually, not really. Cerebro was born out of academic needs that I saw. And the reason for this, and I joke about this in my Cider short talk, you should go check out the 10 minute video, is that many of these so-called cloud whales that Mike Stonebreaker calls them, the Googles, the Amazons, the Facebook's, the Microsoft's of the world, they have this – What

shall I say? The problem of plenty. For them, a lot of this model building tested like, “Oh, it requests more resources? Here. 100 GPUs. Throw at it.” That's sort of the mentality that I've seen from these big tech companies over the last half a decade, especially in the deep learning context. It's like many of their papers, they talk about, “Oh, we train this on 100 GPUs, and it worked this way.” And I look at this and I was like, “99% of the world cannot afford that.” And it’s just unfathomable to me that they would think that that sort of stuff would fly in the rest of the world.

And in Cerebro, our motivation really was born out of how we went about this in academia. And so the public health use case, it was an IRB protected data set, a terabyte size. It was on our campus cluster, the JCI cluster that folks at the Qualcomm Institute operate. It's NSF funded. It's 100 GPU cluster that is shared by the entire campus. So it’s shared by several research groups. Shared by courses, and so on. And so we really have to be mindful of how much resources we are using. We cannot just throw 100 GPUs and hug everything, like some of these big tech companies like to do.

And interestingly, at Microsoft, I've interacted closely with Microsoft in the past, even within these companies, it's not like every team has access to all those resources. They also have shared clusters internally, and every team has assigned certain quotas on how much machines they can use, how much storage they can use, and so on. So this environment really exists everywhere, even at these big tech companies. But what we found from our experience was let's take this terabytes as data set. What was the state of the art for doing model selection at scale? We really saw two different things, two different worlds that are completely disjoint. One was the world of task parallel systems, task parallel systems that come out of ML world, stuff like Ray, stuff like Datamine, stuff like the Google Vizier system, stuff like Dask and Celery, and all of these tools. And task parallel systems essentially do the following. They replicate your entire data set across the workers that you want to use. And so if I were to use eight workers, the one terabyte data set would blow to eight terabytes. And that means I'm eating up storage space for other people on campus. I'm eating up memory. I’m eating up a lot of resources, network resources that are not needed.

At the other extreme, we had this data parallel world like Harvard, which determined users to. And a lot of companies use Harvard, PyTorch. DP uses that sort of mechanisms. They shard the

data. So you don't have to replicate the data. But because they have sharded the data and they operate on one model at a time, the network communication becomes immense. It's almost thousand times, 10,000 times higher than what we find with Cerebro. And that means you're wasting the network, and that bottlenecks a lot of people. This is really what motivated us to come up with this technique of model hopper parallelism in Cerebro that mitigates the cons of both of these worlds. So it's a hybrid of data parallelism and task parallelism.

And that necessity, like they say, right? Necessity is the mother of invention. We invented this idea, because we found it necessary for us. And we found it necessary for a great many other people. A lot of domain scientists, a lot of academics, even a lot of enterprises and tech companies. And I think a lot of people missed this in the tech world because of this problem of plenty. They think they could just throw machines at everything and get away with it. And in turn, it comes back to bite people. Like if you're in the cloud, and you're just throwing machines at it, who is going to benefit from it? It's the cloud vendors. It's the cloud **[inaudible 00:40:42]**, because they can pocket more money from the enterprises.

And so the incentive structure is really a conflict of interest for cloud companies to kind of promote this, “Oh, just throw more machines at it,” idea. I really fundamentally disagree with that both as an academic and as a scientist, because we should use what we need, not what we can lay our hands upon. And so that's the philosophy that drove the innovations in the Cerebro project. The query optimization innovations, the philosophy from the database world that we want to maximize the resource efficiency by looking at the picture holistically and utilizing the resources that we can to the optimal extent possible. So yeah, so I would say, I did not interact with any of these big tech companies on the Cerebro project early on. And this was precisely the reason, because I saw this fundamental divergent worldviews. And I think a lot of the under appreciation of the worldview of importance of resource efficiency is one of the reasons why the Cerebro project took this course, took this direction.

Now, in the last year or two, we've been interacting a lot with VMware, VMware is also a big tech company. But their philosophy seems to be very different from all these other companies that I've seen. VMware really cares about resource efficiency, and energy usage, and so on. And so they have been very supportive of this project. They gave me two gifts. And they have adopted these ideas, and they're putting in front of their enterprise customers, and so on. And I

think that is important. The resource use and the energy footprint of deep learning is a big societal concern. And increasingly, people are aware of the need to cut down these things. And so I think Cerebro project contributes to that direction.

And I'm pretty sure these big tech companies and these cloud whales will eventually come around to this worldview, because if they're going to sell products to enterprises, enterprises are going to demand bills that are lower than what they have today. They're going to demand resource use that is minimal rather than bloated. And so it will eventually come back. And they will also have to think about these sorts of things. We haven't gotten there yet. But I've been talking about the Cerebro project with Microsoft and Google and Amazon as well. They are interested, but it's just not on their roadmap just yet. Because, I think, currently, they are more interested in getting the infrastructure set up for even the basic end-to-end ML pipeline that I'm talking about. And I think that's fair. Usually the progression of tooling is first thing you want to achieve is feasibility, right? You want to build tooling that actually makes things happen. Then you look at usability. Make it easy for you. Then you look at scalability and performance and resource costs and stuff. So that's the progression of software in general. And eventually they will get there.

[00:43:15] JM: Okay, that makes a lot of sense. I'd like to talk also about another project, which is sorting hat. And that's around the problem of data prep. And we talked a little bit about data prep earlier on in our conversation. But tell me about the – Actually, we've done some shows about data preparation. The one that comes to mind most is Snorkel, if listeners are curious about that. But tell me about sorting hat and the domain of data preparation.

[00:43:50] AK: Oh, yeah, absolutely. So yeah, Snorkel is an awesome project. That's from one of my ex-advisors, Chris Ré, who's at Stanford now. And also one of my friends, Alex Ratner, who's at UDUB. Snorkel really focus on programming of the outputs, the labels. But the programming of the inputs, the feature vectors is also important. And that's where a lot of data prep effort goes into as well. And so these are really two complimentary things.

In the data prep effort side, like I said, there is acquisition, there's organization, there's screening, there's different distinct activities. The data scientists often kind of bundle under old giant umbrella of data cleaning. But in the database world, we know there's not just one giant

umbrella. There're several distinct activities with different technical concerns. And in the sorting hat project, one thing we focused on specifically was let's say these AutoML platforms that have started to come about. And the one that I kind of was really inspired by what Salesforce Einstein.

So 2019 or something like that, or was it ’18? I forget. There was a Strata Data Conference, and I was there, and Salesforce Einstein engineers came and gave a talk about how it works. And I was really like, “Well, that's pretty cool.” You can now just upload a CSV file, specify it as target, and then the entire pipeline has automated for us. And so now as a data scientist, I can do that as a baseline. Or you can actually democratize data science itself by allowing people who don't have data science expertise, who cannot afford the data scientists to actually participate in this like, say, take, I don't know, a small city government official in the Midwest or something like that. They cannot afford a 150,000, 250,000 per year data scientist. So they can just upload their city file or whatever to these sorts of tools and get a prediction for, I don't know, sensors and gentrification and that sort of stuff. So that's the vision of these automated ML platforms.

But then I looked behind the scenes. What is going on in these platforms? And I realized, okay, they are doing automation of the raw CSV file being converted into feature vectors. Then the feature vectors, there are some feature engineering, then there's model hyper parameter search and all of that. The last three bits, the feature engineering, algorithm search and hyper parameter tuning have been studied to depth in the data mining and ML worlds. There're tons of papers around it. There are books around this. There are conferences around this. That AutoML part has been well-studied.

The first part was shockingly ill studied. That's what really surprised me. And that's where you convert the raw data into kind of feature representation that you want. And so what we went about doing in the project sorting hat is kind of bringing the scientific lens to that part of the workflow. And we looked at Transmogrify, which Salesforce open sourced. So that's cool. And then we realized that there're a whole slew of other tools that are coming about. TensorFlow extended their TensorFlow data validation. Amazon had AutoGluon. Pandas also does some of this stuff. And there's closed source systems that do this, Cloud AutoML table, SageMaker auto pilot. And we really wanted to zoom-in on that aspect with the data preparation. And in the vision paper, the SIGMOD DIM workshop in 2019, we characterize this part of the workflow, and then

we call it out specific tasks that are technically precise. What the inputs are? Qhat the outputs are. And then we said these tasks are very common. They will have to be tackled by practically all of these tools. And so we said, the central piece that is missing today is not new algorithmic novelty to do the sort of automation or new fancy ML algorithms. A central piece that is missing today's benchmark level data sets to formalize and understand these tasks and to compare how well these tools do on the sort of automation. And so that's why we went about creating the ML Data Prep Zoo.

The focus of the ML Data Prep Zoo, is to create benchmark definitions of these data prep tasks. What is the input? What is the output? They find them precisely. And create label data sets, input-output pairs, and then compare existing tools on how well they do on these tasks. And so we've defined a suite of tasks, gone around giving talks about it. The first task that we define this feature type inference. Looking at the attribute in your database or in your file, can you tell me is it numerical categorical, or is it timestamp, or something else? You would think that's simple, but actually it's pretty non-trivial.

Take, for example, ZIP Code. It stored as an integer in your database or your CSV file. If you use that as a numeric feature for your ML algorithm, you're going to get garbage results. Somebody has to, in the automation code, come in and say it's actually a categorical. You have to recast it. Do one hot encoding or something else. That step turns out to be quite non-trivial.

And so what we did in the sorting hat project is we define this benchmark, we created labeled data sets, we compare these industrial and open source tools against simple machine learning methods that do the type inference tasks. And we found that the machine learning-based type inference beats all of these industrial and open source tools by a huge margin and give state of the art results or type inference accuracy. And that in turn means that the downstream model that you're building the AutoML tool, the accuracy goes up as well, in many cases.

And so we did this comprehensive empirical comparison. We released the dataset. We released our ML models. It's going to appear at SIGMOD this year, the SIGMOD 21 Conference. And so be on the lookout for it. And my takeaway from this whole experience was there's a shocking dearth of benchmark level datasets for understanding automated data preparation and automated ML platforms implications. There is a lot of effort on AutoML heuristics for algorithm

search, hyper parameter tuning, feature engineering. We need a lot more work on the data preparation part to understand how well these tools do. Where do they fail? Why do they fail? How do we improve them? And so that is what our project is focused on. And just as a plug, there's going to be a panel discussion at SIGMOD also this 2021, this summer, focused specifically on this. So I will advertise this very soon.

[00:49:32] JM: So the goal is to create a standard benchmark for automated data prep so that different automated data prep systems can be measured against each other.

[00:49:48] AK: Absolutely. That's the first and fundamental goal, which is kind of inspired by what ImageNet did for computer vision and what the TPC benchmarks did for SQL performance like in the relational database world. Standardized definition of tasks and data sets that people can use to compare their tools to compare against these ML-based baselines that we have, and so on, so that they can learn where they're doing worse and where they can improve. That's the first goal.

Second goal is from a scientific standpoint, characterized formally and precisely, in what way does data prep accuracy matter for the downstream ML accuracy? A lot of data scientists go around doing a lot of data prep steps with some hunch, with some heuristic intuition that if I do X, it will help boost accuracy. Then they go and do it. And then they realize it doesn't have accuracy. And so that's a wastage of effort. That's a wastage of time. And the reason is, fundamentally, people haven't come at it with a scientific lens to understand, “Hey, if I infer the types wrongly, how does it affect the downstream model? How does it affect the bias-variance noise decomposition?” That is a scientific part that we are also studying as part of this effort. So it will also advance the fundamental science of AutoML platforms itself. Because right now, AutoML platforms has all this work in industry, where they are building a bunch of tools and people think, “Oh, they just want to cash-in and make money.” But, actually, we are realizing that AutoML platform system requires some science behind it. And this is one of the fundamental parts of the science behind it. And it involves learning theoretic understanding. We need to simulate and see how the bias-variance noise decomposition is affected by different data prep steps. And that is something that I have some experience with in the past.

If you remember, our last podcast was about machine learning joints, schema management. If you don't do a join, how does that learning theory tell us? How does it affect the accuracy? It's kind of a fork-off of that? Here, if you don't do, say, deduplication of categories. Let's say you have California and CA. I just choose not to duplicate them. How does it affect the bias -variance noise composition? And in what context? Can we come up with decision heuristics? Can we come up with, say, some rules of thumb or recommendation systems that tell us, “This is where you need to invest your time in for data prep. And this is where you will get maximum bang for your buck.” That's sort of the grand vision from the scientific standpoint as well. And the third part is to get industry to actually rally around this and actually start taking this path seriously.

So far, like I said, they've been focused on building out the tooling and getting the feasibility and getting the customers in place. But what is the point of getting customers to adopt your AutoML tool if it is going to give you garbage results or if it's a bait and switch? You promised them that fully automated everything, but then it's actually doing a crappy job of the automation. That's not fair to the customers. And so all these companies need to really think deeply about how their tools are working. And hopefully this benchmark effort will be a step in that direction of getting industry to rally around it and think deeply about doing the right thing for their customers as well. And that's a larger consortium style effort that I'm hoping to start. We'll see how it goes.

So far, Google and Amazon have been collaborating with us. And I'm really excited about this collaboration with the TFX and TFdb team. Google had given me a gift last year. Amazon gave me a gift this year. And we've been talking to all these teams that are interested in this stuff. And I'm looking forward to interactions with more of these AutoML vendors and cloud vendors in the space.

[00:53:05] JM: Well, we could talk about a lot more. You have a wide breadth of things you're working on and interested in. But we're nearing the end of our time. To conclude, I just like to get your perspective for how you see the near future of machine learning and data systems evolving. What are the bottlenecks you see being overcome in the near future and the applications you see being built?

[00:53:33] AK: It's a good question of crystal ball question. I don't have a crystal ball. But one thing I would say is it's really encouraging to see a lot of activity in the space in both not just in industry, but also in academia and research community. We have the MLSys, SystemsML Conference that started a couple years ago. We have the scalable data science track at VLDB that I'm part of, and I've written about both of these recently. KDD has its own applied data science track. So a lot of these communities are now increasingly focused on the scientific and technical issues around this.

In the industrial world, there're a lot of industry conferences that have come about in this space where they are shining a light on the open bottlenecks and problems. I think there is a lot of research attention paid towards problems that are very, very well scoped, like compilers and architecture for deep neural networks. That's a very heavily studied problem. I think that problem will get solved pretty soon, because there's huge amount of industry effort and research effort focused on their. Problems that are wilder, that are bully, that are less well-defined, there will be a lot of research attention that is going to be needed. A lot of industry academia partnership that is needed. And I think this data preparation step is one of those. We need a lot more work on this data preparation and data sourcing part of the lifecycle from the scientific and research standpoint. We also need a lot more work in understanding the overarching process of how do people design and build models, rather than just the specifics of scaling a single model or the compilation or the architectural aspects of the training. And we need a lot more industry academia partnership on the deployment side. For feature stores and ML platforms, how do we define this? Like what is the feature store? How do we standardize this? How do we benchmark them? Those sorts of things, I think there needs to be more industry academia partnership. So this is sort of where I see the world heading in the next five years at least.

[00:55:23] JM: Alright. Well, Arun, thank you so much for your time. It's been a pleasure talking to you.

[00:55:27] AK: Thank you again for having me, Jeff, and looking forward to this and more such interesting articles from your folks at the Software Daily. So thank you again for having me.

[00:55:37] JM: Absolutely.

[END]

