EPISODE 1154

[INTRODUCTION]

[00:00:00] JM: Federated learning is machine learning without a centralized data source. Federated learning enables mobile phones or edge servers to collaboratively learn a shared prediction model while keeping all of the training data on device. Mike Lee Williams is an expert in federated learning and he joins the show to give an overview of the subject and share his thoughts on its applications.

[INTERVIEW]

[00:00:27] JM: Mike, welcome to the show.

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

[00:00:30] JM: I want to start by talking about some privacy concerns around machine learning. What kinds of information in machine learning is considered sensitive? Are there laws around what kinds of features should be kept private?

[00:00:44] MLW: Yes. Yeah. I mean, that is the sure answer to the question. I mean, I think probably the example that we're all most familiar with or that is most salient for many of us is photos. Many of us upload our photos to the cloud these days and they disappear to most of us. We know not where. And machine learning had done on them and other kinds of processing tasks had done of them. And I think for many us, that creates at least at the back of our mind a feeling of discomfort. But more concretely, there is of course medical data. A lot of our healthcare data is recorded digitally and potentially gets transferred between different entities, different healthcare providers. And in the US, there's HIPAA, elsewhere, their equivalent laws that constraint the ways in which that data can be managed, auditing access to it, these sorts of things.

But the big like 90-pound gorilla of regulation in this area that really only arrived in, I think, it was May 2019. So not that long ago now, is of course the European Union's GDPR. And that has kind of changed everything and changed nothing in some ways. It's not really clear how it's

going to enforced yet. So I'm going out on a bit of a limb here when I talk about what its implications have been. But generally what that law creates is a legal version of an idea I think most of us understand when we talk about data, which data is risky and you are under sort of an obligation to the source of the data when you become a custodian of that data. So you want to minimize the amount of data you collect. You want to ideally not store it for any longer than you have to. And you want to act on the source of the data's preferences. If they say, "Can you delete that data?" Then yeah, you better delete that data.

And those kinds of ideas are enshrined in GDPR, and the GDPR applies to anyone who's a resident in the European Union, which is very, very large group of people, 400 million people or whatever it is. So it's very difficult to conduct like an internet-scale business without being affected by GDPR. And California as of I think January of this year has a very similar law. So that's kind of the environmental, the legal environment in which we're living. But at least half of the problem from the point of view of someone who wants to do machine learning with data is the preferences of their users, which are may not be legally binding. But if you want customers, you need to think about their preferences. So that's sort of the environment we're working in.

[00:03:17] JM: So one simple but effective way is to mask information. Can you explain what masking means?

[00:03:24] MLW: Sure. The obvious thing to do if there's a piece of information that you should not have is simply to not collect it or sensor it from your data. So let's say you're doing – It's the election season right now when we're recording. Let's say you're doing a survey. You might choose not to record specific pieces of information such as gender, race, ZIP code, things like this. And in that way, you plug the most obvious security privacy hole in the data, which is that you have the data.

That only gets you part of the way though, because it's quite often possible to infer, to fill in the blanks if you like from this masked information. So an example, I could ask a lot of questions about you and choose not to record your ZIP code, to mask that piece of information out in my dataset. But I may ask enough questions about you to make you unique in your ZIP code. And I can invert the data I have and figure out where you live based on the other answers you've given me.

So masking is like a very – Accrued is perhaps not the right word, but it's a simple approach that was often the place to start, but is not sufficient. Of course, you lose the information. If you do your masking well, then you do not have access to information that is potentially useful for whatever downstream task I want to do. So if I want to learn about the distribution of income in the United States and I choose not to record people's ZIP code, then clearly I am strong in that analysis. I can't do a good job, because I don't know where people live. So what we'd like really is some kind of way to record that information in a way that makes it accessible to analysis while keeping it at arm's length if you like.

[00:05:10] JM: Could you define the term differential privacy?

[00:05:13] MLW: Yes. This is one of my favorite subjects, differential privacy. So differential privacy is an extremely formal technical field. By the standards of machine learning, it's actually a pretty mature field as well. The true math and statistics nerds go to different privacy talks at conferences. It's a very rigorous field compared to the rest of machine learning, which is the Wild West. The basic idea however, like having said that which might have you put you off. The basic idea actually is pretty simple. So let's say, again, back to this income example. I'm in a room with 50 people and I want to know the average income of all the people in the room, and most of these people quite reasonably not willing to tell me their income.

What we then do or the differential privacy approach to solving this problem is to ask everyone in the room to pull up Python or whatever and generate a random number with a min of zero normally distributed plus or minus $10,000 or something to their salary. So they take their true salary and they perturb it by some random number, and they tell me that perturb value. And then I combine all the 50 values from the people in the room and take the average of those 50 perturbed valued. And the nice thing about this is I didn't actually see any one's naked salary, if you like, anyone's unperturbed salary.

Now in the case of this example, I can probably – I now know in the ballpark of what your salary is. But the basic idea that I didn't find out the true answer, true values there. But despite adding all these noise to the data, it's a statistical property of the normal distribution, the bell curve, that when I take the average of these values I recover on average the true average. So I do indeed

get a good idea of what the average salary is in the room. That's differential privacy. And there are limits to this. If there's a categorical variable, I can't ask people to – I don't know, perturb whether they're left-handed or right-handed. You're one or the other. So the detail – You can't add a normal distribution to your handedness.

So the details of how you perturb the data vary a lot, and the implications of that perturbation for how accurately you can recover the information of that's how the sausages are made. That's the field of differential privacy. But the basic idea is simply add noise to the data such that the aggregate value you're interested in, the average or whatever it is, is still correct.

[00:07:48] JM: When people think about privacy concerns with big data, they often think of data breaches and direct access to data. But this is not the only way to get information. Explain how an attacker can get information from a trained model without accessing the underlying data.

[00:08:06] MLW: Yes. So this is the new frontier I think in a lot of ways. Machine learning models are in some ways really not much more than lossily compressed representations of the original data. So if you think about you're in high school and you are measuring the length of a spring and putting different weights on it and you plot the length of the spring and how much weight you put on it, and you put a little axis on your chart. That's your original data. And then in high school where you draw a straight line through that data and recovered, Hooke's law or whatever it is, that's basically machine learning. You are summarizing the hundreds of numbers you recorded with a couple of numbers. It's just lossy compression in that sense. You capture the broad brush behavior of the data while compressing the data.

And just like lossy compression, a lossily compressed version of a photograph of me, I mean, it depends on how lossily you compress it. But you can probably still figure out what I look like from that photo. The details of how this can be done with machine learning, it varies from completely trivial attacks, where it's obvious what was going on in the original data, to very, very challenging situations. And generally speaking, it's easier for simpler machine learning models to recover the training data or invert the model is the fancy way of saying it. So in that situation, with the spring and the weights, it might be quite easy to figure out what was going on with the original data.

Another example where it might be quite easy is what's called a bag of words model, a language model, where each word has a parameter. So the word car has the number 3.5 in the word. Garage has the number 4.8, whatever. A machine learning model which goes through training and then the weight of a particular word changes, you can pretty much guess that that word occurred in the original data. So if the weight of the word garage changes, then probably the text I trained it on had the word garage in it.

The evolutionary tree of machine learning, we're now at the level where even the people building the models don't understand them. And in that situation, inverting a model or figuring out what was going on the training data is much harder. No question. There are good reasons to want to do that for not just security reasons. I'm an attacker and I want to figure out what's on in the training data. But there's also the issue of machine learning interpretability, like here's a problem if we don't know how machine learnig models work. And these kinds of attacks also – Not attacks. They're actually reasonable good things to do to interrogate what's going on with your model.

So there's a lot of work going on in this area. But the basic takeaway is a machine learning model is just an echo, a compressed version of the original data and that has implications for how well hidden the original data is in the model, and not vary is the basic idea.

[00:11:08] JM: What are the most common ways of providing differential privacy?

[00:11:12] MLW: So we kind of touched on like the simplest example is I want to add Gaussian noise and normal distribution there about 500 different ways of saying a normal distribution. I'm sorry. I'm switching between them. Gaussian noise, normal distribution, bell curve, they're all the same thing. That's the most obvious thing to do when adding noise to continuous data, by which I mean numbers. And that's often what we're dealing with. You can apply that kind of noise to an image. You can apply it to tabular data, someone's blood pressure, height, weight, all these things. And that works fine.

It becomes quite challenging however when you got – The fancy way of saying it is categorical data. Data where there's a finite set of – There's an enumeration of possible values. And the way you perturb that data is generally a trickier problem to prove things are going to work out

okay, to prove the data will not be recoverable and to prove the downstream analysis is going to work, is going to recover approximately the right answer despite adding the noise.

I think that's kind of what I can really – I'm a little bit out of my wheelhouse on differential privacy. So probably the only – Like the extent to which I can get concrete about the approaches. The use cases, I can talk about a little bit, and that might be at least as useful. I think the most prominent user of different privacy on earth right now is Apple. They've made this like a selling point, that they do not have access to your data, and there are two ways in which they're doing. One is they apply machine learning models on your device. So they want to make a prediction, a categorization of your photos or whatever. And they apply the model on the device. So they don't transfer the data to some high-powered server and apply the machine learning model and then give you back the answer. They have you do it on your very powerful, these days, phone.

But the other way they do this is, the other way they avoid touching your data is differential privacy, and they do this particularly, my understanding at least, I don't work at Apple. And of course they're not famous for me, open with what they're doing. But I think the most prominent way in which they're applied this stuff is to metrics of usage. So operating system, metrics, and they make clear that they are dithering, adding noise to this data and hopefully providing warm, fuzzy feeling for the user that they are contributing to making the software better whilst also not revealing their own patterns of usage.

[00:13:39] JM: Let's go ahead and get into the topic of federated learning. Could you define the term federated learning?

[00:13:44] MLW: Sure. So already touched so far on what machine learning is an the kinds of problems we're worried about, about access to data. In general, machine learning models are better when they have access to more data. But there are all these reasons why we can't or don't have access to this lovely data. Of course, privacy is in some ways the big one these days for all sorts of very good political reasons. The other reason we might not have access to this data is just good old fashion engineering constraints. The data stats' life is born on born on some remote device, a phone, an IoT, sensor in the jungle, a mine. Who knows where? It's born in some far-flung location. And the very act of getting it back to some centralized server, a data

center, is challenging. We just don't have the resource to do that in terms of bandwidth or batter power. Maybe we can get it back, but now we have another problem, which is we need hard drives to store all these stuff on. Potentially, it's a very large amount of data.

So federated learning, the goal of federated learning is to get around this issue that having access to data is sometimes impossible because of privacy concerns and sometimes impossible or undesirable because of engineering constraints. And the algorithm is pretty simple. It's a zoo of algorithms, and I'm going to describe the simplest one, I think, which is called federated averaging.

So the world starts like this. I'm a server and I have a bunch of phones that are acquiring data, and I'd like to train a machine learning model on that data. So what I do as the server is I send an instruction to all these phones saying, "Can you train a model that is of this type on the data that you have? Don't talk to each other. Don't tell me about the data. Just train a model on the data you have."

Each of these phones doesn't have a lot of data. So the individual models are not great. They might capture the specific patterns on that phone, but they're not going to capture the great variation that exists elsewhere. But here's the clever bit. Instead of having them send me as a server the data back, I have them send the trained model back. And then the clue is in the name, federated averaging. What I do with those models that I got back from the phone is the most obvious thing you could think of to do, which is take the average of the models. And if I take the average, I mean literally. I find the first parameter. I find its value in every model I've been sent back, and I take the average. And maybe I do something clever, like weight your phone more, Jeff, because you had more training data than my phone, for example.

But the basic idea is I'd take the average. And now I've got a combined model that in some ways captures all the data that was present on all the phones. And we're not done here. We'll so several rounds of this. So I'll send this average model back to the phones and say, "Could you train again with some new data? And instead of starting from scratch, start from this model I just sent you." And that's a round of federated learning, and we'll do this many times, and there's devil in the detail about when to do it and how to do it such that we don't waste your batter or your bandwidth and so on. Maybe we should get into that. But the basic idea of doing several

rounds of this training leaves us in a state, us here being the server in all the phones, where we all have the same model, has been trained effectively on all the data, the union of the data on all the phones. But crucially, none of us had to share, directly share our data, and that plugs the most obvious hole in machine learning, which is in a very literal sense I had to send – I had to share my data.

And also it saves power and bandwidth, because as we mentioned, a machine learning model is a compressed representation of the data potentially by hundreds or thousands, or tens of thousands of times. Smaller than the source data, which means I'm sending as a phone less data around. And that's significant not just because bandwidth is expensive, but because particularly in the case of phones, using the antenna is the most demanding in terms of battery life as well. It depends on what you're doing, but it can be more demanding than having the screen on. So we want to minimize the amount of data we're sending around from that engineering constraint point of view. So federated learning really addresses to a great or a lesser extent. And we've already kind of touched on some of the ways in which this isn't going to completely solve the problem on these concerns of privacy and engineering resources.

[00:18:16] JM: Federated learning was first coined and used by Google. Could you explain the setting in which federated learning was first developed?

[00:18:25] MLW: Yes. Yeah. Google's use case of federated learning is in some ways the perfect to most obvious use cases. It's not a surprise it came out of Google. So they are of course the maker of android, and they are also a company that for better or worse has a reputation as an ad tech company being potentially a little bit cavalier about privacy.

Federated learning is a great way of addressing that, and I don't just mean its PI. It really is like a rigorous great way of addressing that. They're a little bit caged by exactly what they're using it for. But the use case they've said the most about is one we talked about in a little bit detail, which is Gboard, the predictive Google keyboard. So I don't have an android phone. So I'm maybe mangling some of the nouns here. But the basic idea is you write something and it predicts the next word, and that way you can type more quickly.

Now the problem – This model works with machine learning, of course. And the problem with

that is in order to train the model, you've got to share everything you're typing. Essentially, you go and install a key logger. For very obvious reasons, people don't want to do that. And federated learning gets around that by having people incrementally train a shared model with pulled resources where the model is combined into an average.

Federated learning works particularly well in this situation for a couple of reasons. One is that the user of the phone generates training data without trying. It's just like part of the exhaust of like the – A waste product, if you like, of using a phone, is you generate training data for this model whether you like it or not. It's completely passively generated a huge volume. So there's a lot of training data out there for Google, and it's very, very much higher quality than the training data they would get by, for example, downloading the full text of Wikipedia, which is written in generally pretty good English compared to, for example, text messages. But it is not how people write text messages. So it's not going to result in good prediction. So this is the other way in which this is a good setting.

By using this technique, Google and people using the technique elsewhere can very quickly pick up changes in language, unique features of language that are particular to a specific location or a specific time, like a new word that people have started saying on Twitter or whatever. It's going to show up in their model rather more quickly than it would if they were once a year batch training on some dataset they grabbed together from Wikipedia or something like that, which is what the rest of us mortals have to do who do not have access to Google's user base.

[00:21:06] JM: So how have the opportunities or the necessary fields of federated learning increased in recent years?

[00:21:15] MLW: So yeah, that's a great question. We've sort of emphasized privacy as the issue that federated learning addresses, and it is. And that makes most of us think about our own data. But of course there are much larger entities that have privacy concerns than just me and my baby photos or you and your salary or whatever. There are hospitals, drug and medication research, entities. These are very, very secretive organizations. They also have huge legal responsibilities around the data. There are insurance companies that whose data is essentially how they make their money, is that their training data around rare events is incredibly valuable.

So one of the ways in which federated learning has evolved over the last few years, I think the federated averaging Google paper was 2015, 2016, something like. As a field, it's about – It depends how you count. Ideas like these have been around for 20 or 30 years. But the branding as federated learning is about 5-years-old. The changes in those 5 years really have been – Is application to new fields, like insurance and healthcare. So outside of the traditional interests of technology companies. These are very, very high-stakes industries. A lot of money involved, and also potentially the opportunity to make really positive impacts on people's lives with things like cancer treatments and stuff like that.

And then the other area of progress active research is really what we talked about in the start. These concerns that you can recover the training data even from the model. And that's true of federated learning. It's a pretty hard problem on the spectrum of like machine learning attacks. But it is possible. And if you are doing something like using federated learning to protect healthcare data, then you better make sure it's impossible. And guarantees around that with things like differential privacy and a secure aggregation, which we haven't talked about, the other area of really active work right now.

[00:23:10] JM: Tell me how a model gets updated in a federated learning process.

[00:23:15] MLW: Let's go back to that example that I'm a server and there are 5 or so phones, N phones connected to me, and we are talking to each other in rounds. So I'm saying, "Can you train a model?" And you send me an update, model trained on your data, and then I take the average of all the phones and send it back and say, "Can you train again starting from this data?" That's the very high-level description of what's going on in federated averaging at least on the other versions of federated learning. Their essence is not terribly different from that.

We skipped a lot of detail here, which if any of your listeners are distributed systems people, their spider sense will be tingling, like we've lighted a bunch of complexity there that is very, very familiar to distributed systems people. The couple of ones that are worth touching on, particularly in the context of phones, they are stragglers and drop connections. What about these phones that – I've got an iPhone 6s. You've got an iPhone 11. Your phone is going to train quicker than mine. But because of the algorithm I described, the algorithm is only going to

proceed as the fastest or the slowest phone, because it's going to wait for all the phones to phone home with their updated model. And that is generally speaking not satisfactory. And the worst case scenario is what if I shot my phone off? Now everyone is stuck waiting for me to turn it back on.

So solutions to those kinds of distribution system problems that you're dealing with a heterogeneous environment and you're dealing with Murphy's law. Something is going to go wrong. And especially if N is large, everything is going to go wrong eventually. These are new problems for most of us in the machine learning community. We haven't done a ton of thinking about them. We don't know a lot about them. So the joining of the distributed systems community and the machine learning community and attacking these problems I think is super interesting.

The basic idea though is these rounds of communication where you take averages. There's one more sort of improvement you can make that I want to touch on, which is I mentioned that the model is a compressed version of the data, and it is. But can we compress it even more? Or can we minimize the amount of bandwidth we're sending even more? Not just with privacy in mind, but also with saving bandwidth and battery in mind.

And there are a number of strategies for doing that. Perhaps one that's receiving the most attention right now is a technique called quantization. So it's taking a machine learning model as normally a very large number of 32 bit flows. So hundreds of megabytes potentially for neural network, gigabytes even. Quantization is saying, "Right. We're going to turn that model into integers, or maybe even into Booleans, ones and zeroes. How much can we get away with without breaking the model?" And it turns out you can get away with quite a lot. And obviously that saves a huge, huge amount of bandwidth when we start sending these models around on phone connections.

[00:26:04] JM: Are there particular types of models that can be trained with federated learning? Or is it just deep models? Or any kinds of model can be trained with federated learning?

[00:26:14] MLW: Thank you for asking this question. This is I think one of the misconceptions about federated learning, because it's seen as quite like a sexy modern machine learning

technology. There's a sense in which sexy modern machine learning is synonymous with deep learning. And for the purposes of federated learning, deep learning is not special. The only requirement is that you can meaningfully take the average of two trained machine learning models. So that's actually much simpler to do for something like a linear model. A liner model is just going to be a list of numbers. Back to the springs and the weights example, the linear model of that data is probably going to be two numbers, the gradient and the intercept.

And it's very obvious how you would take the average of two of those models. You take the average of the gradients and the average of the intercepts. For neural network the idea is to say every – There's a very large number of parameters. They have some kind of structure, and you take the element-wise average of those parameters. And if this was not a podcast, but a conversation, you'd see me waving my hands right now. And I'm not sure if that would add anything. But it's a little bit hard to get your head around, but it's the same basic idea as to what's going on with a linear model.

There are a couple of fairly popular approaches to machine learning that cannot be averaged in this way, and the most prominent of those are trees and forests, a decision tree or a random forest. And the reason for that is if I trained a decision tree on one set of data and you trained a decision tree on data with the same features, but different data, then we're not just going to come up with a different model. We're going to come up with two models with different shapes, different structures. And that means there's no really well-posed way of taking the average of our two models.

So with the caveat that you can't apply federated averaging at least to decision trees, random forests, gradient boosted decision trees. So all of these are very popular approaches in ad tech and things like that. In most other situations, you're going to apply. And in particular, linear models, SVMs, deep learning, they're all fine. Oh! I should add, there is one constraint, one additional constraints however, which is that you're training these models on what can be in some cases pretty esoteric hardware. So a phone, an Arduino, some weird IoT sensor, and that presents just practical architecture problems. It might be tricky to get TensorFlow or Torch running on those systems. And TensorFlow or Torch or whatever you're using doesn't just have to run on those systems in the sense of inference. We're talking real machine learning training on those devices, and those are not problems that have been solved everywhere.

[00:28:55] JM: How can we be sure that a model trained using federated learning performs comparatively well to a model that would be trained form a central data source?

[00:29:03] MLW: Yeah, that's a tricky one. In general, federated learning is a hustle. Ideally, you wouldn't go to all these trouble. You would just put all the data on one computer and train it there. And of course, you don't do that with federated learning, because you can't, which makes a pretty unlikely situation where you can compare the performance of your federated model to the performance of the model you would get if pulled all the data in one place. But certainly, the academics who have studied this have made sure to run these tests. And in some cases you can actually prove it mathematically that the federated model converges on the same performance as a model where all the data is consolidated in the data center.

Generally speaking, with a linear model, you can do that proof. With a neural network, the problem with proving anything about a neural network, and federated learning isn't special here, is it's very difficult to make strong statements about why neural networks work when they're going to work. It's very empirical. You give it a try and it results in improved performance. And that's certainly true in the case of federated learning.

So the best way I can do is say in situations where we have fake data that we have the opportunity to pull in one place, it performed somewhat better. The way I got into federated learning was running exactly these tests myself trying to verify if these claims are in fact true, because in the case of a linear model, it's clear that it would work. But in the case of a neural network, it's not obvious – I mean, it's kind of surprising that it works. I think it's fair to say in the case of a neural network.

So Cloudera built a simulator essentially to simulate federated learning training in industrial predictive maintenance data, and we were able to show that it does indeed result in improved performance. And the institution is clear. The reason that federated learning model is better than just training on your subset of the data is that you have access to more data. But you can think of these rounds of communication as lossy as well. So federated learning has a strict upper limit, and in some cases it won't reach that limit, that it can't do any better than putting all the data in one place. Certainly, from an engineering point, I view it as a tremendous hustle

compared to putting all the data in one place. So if that option is available to you, if you don't care about privacy. If you've got big enough hard drives and fast enough internet connections, you should put all the data in one place.

[00:31:32] JM: How practical it is to build federated learning? Is it very difficult to build, or are there good tools for building it today?

[00:31:40] MLW: So federated learning is kind of at the intersection of a really horrible Venn diagram of cryptography, different privacy, distributed systems, machine learning. These are all like very, very tricky areas to work. Federated learning is in the middle of all of them.

So I'm afraid to say it remains pretty tricky. There are a couple of off-the-shelf open source approaches that I want to flag up. TensorFlow federated and PySIFT. TensorFlow federated is what you'd expect. It's a subset of what Google uses internally. If I understand correctly, I'm not a TensorFlow user, so I apologize to this team if I'm mangling at this description. But the open source release is missing the secure aggregation part. And the secure aggregation, we skipped over, but it's the basic idea of formal guarantees that the server cannot read in plain text the model, the nodes send back.

And Google certainly, one would hope, and as far as we know is using that stuff internally, but it's not part of the open source TensorFlow federated suite. PySIFT is an independent project that attempts to – Does include a secure aggregation component. It also addresses – It begins to address the networking issues, the distributed systems issues I mentioned about stragglers and drop connections and things like that. It's a very fast-moving, energetic, if you like, open source project, which makes it a little bit hard to keep up with. And one caveat about PySIFT that they have right there on the GitHub page is if you care about privacy, don't use this in production yet. These guarantees haven't been verified. There are probably bugs, I assume, is what they're getting out there with that warning.

So the two options, in some ways, neither of them is ready for production yet unless you are the likes of Google. So TensorFlow federated the open source, but is missing a key piece of the puzzle. And PySIFT, it sounds like pre-1.0 open source software. So the situation is in that sense a little bit unsatisfactory. If you don't care about privacy as much, if the problem you're

trying to solve is compression, saving data, saving power, the situation is much better. There the stakes are much lower. You can only screw up so much, and you notice if you screwed up more quickly than you do when you screw up privacy. So there, there are more off-the-shelf tools. And anything that can handle network I/O is going to get you up and running, plus a machine learning library.

[00:34:17] JM: And what about other practical problems with federated learning? For example, if I have a very large model and I want to send this model to a phone, is that going to be a problem? Is it a problem to get the bandwidth and the storage and the CPU concerns? Are there some restrictions involved here?

[00:34:35] MLW: Yeah. I mean, at the very basic level, there's an idea that what's going on on your phone is obscure to most owners of the phone. I don't understand what my phone does, and I do machine learning for a living. So this idea that you're asking consent here to do this kind of thing on people's phones is a little bit slippery I think. None of us read these agreements. And embedded in this agreement is not just that we're going to do something that if we screw up, it's going to compromise your privacy. And even if we don't screw up, it's going to cost you batter is an interesting one.

What most production phone deployments do is only train like roughly once a day or when it's on main's power, when it's plugged in. And that's for the obvious reason, using the CPU or if it has a GPU on the phone, it's going to be quite batter intensive, setting aside the issue of communication. Communication, we've already touched on a few other ways in which you can in many ways bandwidth concerns. And some of these concerns, I mean it depends where you are in the world how acute these concerns are on a cellphone. Like the US where we are, is infamously expensive for bandwidth, even relative to the cost of living is very high. And it's surprisingly cheap in some parts of the world, like Europe is half the price, but then there are bits of sub-Saharan Africa where it's pennies for a gigabyte bandwidth. Local context is relevant here to how much you are about compression and data. But we've touched on the ways of doing that. Quantization is one of them. There are twists you can do to the communication protocol as well. And not just quantization, but having a phone sit out a few rounds. Basically, if it's trained the model and it kind of hasn't changed much, then there's really not much point in uploading it again. So just have it sit out until it's taken enough photos or sent enough text

messages to change the model significantly. Yeah, that's it for practical issues I think that I've got right now.

[00:36:33] JM: We talked about privacy earlier, and now we're talking about federated learning. Help me bridge the gap between those two a little bit more. How does federated learning achieve differential privacy?

[00:36:44] MLW: Yup. So federated learning, by keeping all the data on the nodes solves the most glaring security hole in your data, which is that you have to give the data in plane text to someone else. That no longer happens in federated learning. Vanilla federated learning stops there though. It does not address the problem that we talked about also at the start, which is that in principle, you can invert a machine learning model. You can look carefully a machine learning model and figure out things that were true of a training data. That's simply again because it's a compressed version of the training data in some ways.

Federated learning isn't going to help with that. You are in general protected from other nodes in the network, and that's because all communication, your model is only ever shown to the server. So the server is the vulnerable point, if you like, for that attack of model inversion. But really, for production use of federated learning, when you care about privacy either because you legally have to, because your customers have very strong preferences all because you just don't want to paint a target on your back and have this data, then there are extra steps you need to take, and the options really are different privacy or secure aggregation. Differential privacy, we kind of talked about. It's the idea of adding noise to the data. And the example I use was adding noise to the training data, if you like. I said add noise to your salary and then you can take the average. But it's nothing to stop you adding noise to the machine learning model. So machine learning model is just a list of numbers. Add some noise to it. Upload it and then it becomes harder to invest the model.

The other option you have is secure aggregation. So secure aggregation is one version of multiparty computation. It's this idea that we can come up with ways of changing the algebraic operations that are performed. That means we recover the right answer, but we don't actually ever see the plain text ingredients to that calculation. That probably didn't make sense. So let me back up to like 2+2=4.

Say I wanted to add two numbers together, but I didn't want to reveal what the two numbers are. I just want you to be able to recover the right answer. What I might do – If there's just two of us, we can't do it. But if there was three of us, we could do it. What I would do is take my number. It might be my salary, and add some very large number to it, just a constant very large number to it. Pass it on to the next person in the line. They will add their salary to the sum of my salary and that large number and then they'll pass it back to you and you'll add your salary. And then you'll pass it back to me, and I can subtract that very big number that I originally added. And now I know exactly. This is not differential privacy anymore. There's no noise here. I know exactly what the sum of our salaries is. And none of the three of us could recover anyone else's salary in that process. That's an example of multiparty computation. You can do a version of that with federated learning called secure aggregation, and it's a big different because the server is special. In multiparty computation, I just subscribed those three people were peers in some ways. But a sever in federated learning is a big special. So you need – Just adding constants won't work. You need to do something called hemimorphic encryption, which encrypts the numbers such that the basic algebraic operations still work.

I take the encrypted version of two and the encrypted version of two and add them together and get an encrypted version of four, and I decrypt it, and I recover four. It depends how I encrypt the data. If I encrypt the data with MD5, that idea is not going to work. Addition doesn't work on MD5 encrypted data. But there are encryption algorithms for which that will work. So to come back to your question, I said a lot about secure aggregation, which is kind of not off topic, but probably more than you wanted. The idea is federated learning plugs the first and biggest and worst hole, which is that you share the data in plain text. But if privacy is the concern, then there are additional steps you might want to consider, differential privacy and secure aggregation.

[00:41:03] JM: Do you see many people in the industry using federated learning? Have you seen many practical applications of it or do you think it's still too hard to implement even with the tooling that is out there?

[00:41:16] MLW: So certain Google are using it. I was going to say they're very public. They're fairly public about what they're doing with it, and that's to their credit. There's been some research of other companies that could in principle have applications for this. So there's a

Samsung paper, for example, using this. And you can think of applications for this. But the other big use of federated learning – Well, in production is probably the wrong word, because these aren't really technology companies using it in that live production systems, but it's being used to solve like batch problems in healthcare, drug discovery. So we mentioned right at the start that health data is like extremely sensitive. They have ethical and like legal constraints on what you do with a data.

And carefully implemented, federated learning plus differential privacy handle secure aggregation would seem to get around those issues and allow, for example, two hospitals or many hospitals or many drug companies to pull their training data and discover lifesaving drugs more quickly. Now, I'm not super familiar with how this work is being conducted in the US, and I don't know if this claim is true, but it wouldn't surprise me to learn given the political climate around healthcare and the US that it wasn't happening, but it is happening in Europe. The European Union called for proposals a couple of years ago and then gave out a bunch of the money to have people set up essentially these federations in the legal sense that you can join. And then the federation comes and literally puts a PC in your office, like some hardware in your office that is the node of a federated learning network. So there's like a zoo of startups that have come up around this healthcare ecosystem, in Europe in particular.

Owkin is perhaps the furthest along of those, but there are a couple of – And they've established a thing called the Substra Foundation. And then there's a thing called Vantage6 as well, and these are somehow connected to the European Union, which always has the consequence of making them a little bit difficult to read. But they are working on healthcare on the European Union.

There are a couple of startups as well who in one case appears to be trying to build a platform for this. You as a company whose core competency is not machine learning or federated learning can ship federated learning either into production, and you can pool with other companies who might be your competitors or you might constrained of showing data with and be mediated by these companies. The two I know about are Cape Privacy and Datafleets. Cape Privacy is interesting, because they're doing the work to publicly fill in the gap. I mentioned in the coverage of TensorFlow federated. TensorFlow federated doesn't have an open source implementation of encrypted machine learning that I know of, like, wired into TensorFlow

federated. And Cape Privacy are doing the work in open source to fill that gap in, which is great.

[00:44:21] JM: As we begin to close off, let's talk a little bit about where you work, which is Cloudera, I believe?

[00:44:28] MLW: That's right. That's right. I work at Cloudera, which my bosses will kill me for saying this, but is best known as a Hadoop vendor. And I don't know that this – I suspect that's where we make most of our money as well. We're not known as a machine learning company. I came in through an acquisition of a little research lab called FastForward Labs, which Hilary Mason started in Brooklyn a few years ago, and we got acquired and absorbed into the mothership.

And the business model of FastForward Labs is kind of weird and interesting. We release white papers, and originally we release them to very well-paying customers who got a nice a physical book and read them. And since we've been acquired, we've stayed giving more of this stuff away. So the federated learning work I did, which it will take you an hour to read if you're interested. You can see at federated.fastforwardlabs.com, and it's a report explaining some of the things I've said. And in some cases, Cloudera customers are sufficiently far along in their journey towards like machine learning. They're interested in the plain technologies like federated learning in production. So we have helped in a couple of cases or scope how that would be done and figure out when it would work and when it wouldn't.

[00:45:45] JM: Do you know much about what's on the cutting edge of federated learning? Google was obviously earlier to the federated learning game. What kinds of cutting edge developments would a company like Google be working on?

[00:45:57] MLW: Yeah. We've touched on the issue of like privacy guarantees. Tying the service hands to make it logically impossible for anyone to invert the data. That's certainly probably the area of most active research. It's also where the money is going to be made in this field, because it opens up healthcare to an extent. And it's not currently open.

The other area I think is worth touching on is customizing federated learning models at the user level, at the node level. So if I'm participating in the Google keyboard Gboard model and having

it predict what I'm going to write next, in some ways I benefit from pulling my data or indirectly pulling my data via federated learning with lots of other people. But you can imagine a situation where that would make my model worst. I'm British, and I assume may be able to get -. I'm British and I live in the US. So if you pull my language with the language of my neighbors, then the small amount of training data I contribute is going to be overwhelmed by people who speak very differently to me. And that may actually make my experience worst.

So customizing federated learning, allowing me to benefit from the broadly defined English that is used all over the world, but allowing it to give more weight, if you like, to my own training data, because I use an S instead of a Z or whatever, or I say zed. That kind of thing is I think a really interesting area in which consumer federated learning is going to get much, much more interesting.

The other area I think – I think it's wrong to call this research, because this is something that like hobbyist can and now are working on. So it's not a research and that you need to go to a conference to work on this stuff, but is applying federated learning on commodity very, very low-priced hardware, like I've been playing around with this a big myself. I gave myself – For quarantine, I gave myself a Raspberry Pi W, which is one of these Raspberry Pi's that costs $10. Like a real computer for $10. It's completely insane. It's slow. But I've got federated learning running on it, and I've got a cluster of one nodes. So I'm not doing anything very interesting with it. It's really just a proof of concept. But this sort of esoteric hardware that hobbyists have access to because of its cheap price is another area I see a lot of really interesting work going on. And that's true of machine learning generally. But federated learning, because it's a way of pulling that kind of hardware is particularly interesting application.

[00:48:23] JM: Cool. Well, this sounds like a great place to close off. Mike, thanks for coming on the show. It's been great talking to you.

[00:48:29] MLW: My pleasure. Thanks very much for having me, Jeff.

[END]

