Speaker 1 ([00:00](https://www.happyscribe.com/transcriptions/f73ad59d129749f4a93b7e653d4b4d4e/edit?onboarding=true&position=0.17&utm_source=happyscribe&utm_medium=document_deep_link&utm_campaign=editor_copy_section&utm_content=f73ad59d129749f4a93b7e653d4b4d4e))  
Your mobile phone has a plethora of rich data to train machine learning models on. Think of predictive text on your keyboard, or the wakeup word to trigger Siri and Google Assistant, or even the healthcare data tracked by the sensors on your phone. So how do Google and Apple Train on such rich but private data while still maintaining your privacy? Through a technique called Federated learning. I'm Muckle, a master's student at the University of Cambridge, and in this video we're going to be diving deep into how federated learning works.

Speaker 1 ([00:32](https://www.happyscribe.com/transcriptions/f73ad59d129749f4a93b7e653d4b4d4e/edit?onboarding=true&position=32.04&utm_source=happyscribe&utm_medium=document_deep_link&utm_campaign=editor_copy_section&utm_content=f73ad59d129749f4a93b7e653d4b4d4e))  
This video is structured as follows. First we'll look at how Federated learning works, and then we'll look at the classic Federated Learning algorithm, Fed average. We'll then look at improvements to fed average and the current state of Federated Learning Research. We'll then look at how you can implement federated learning in practice through frameworks and federated datasets. We'll end by looking at how Federated learning fits into our privacy preserving machine learning toolkit.

Speaker 1 ([01:02](https://www.happyscribe.com/transcriptions/f73ad59d129749f4a93b7e653d4b4d4e/edit?onboarding=true&position=62.01&utm_source=happyscribe&utm_medium=document_deep_link&utm_campaign=editor_copy_section&utm_content=f73ad59d129749f4a93b7e653d4b4d4e))  
Let's get started. Normally, to train a machine learning model, you host both the model and the data on the same device, and we call this centralized machine learning. However, for us that means Apple and Google upload our private conversations to the cloud to train their machine learning models. Federated learning flips the paradigm. Instead of sending our data to the cloud, we send the models to our devices, and then we train these models locally on our devices.

Speaker 1 ([01:29](https://www.happyscribe.com/transcriptions/f73ad59d129749f4a93b7e653d4b4d4e/edit?onboarding=true&position=89.2&utm_source=happyscribe&utm_medium=document_deep_link&utm_campaign=editor_copy_section&utm_content=f73ad59d129749f4a93b7e653d4b4d4e))  
This means the data never leaves our device. Once we've trained our model locally on the device, then rather than sending data to the server, we send the model updates to the server, and the server aggregates the model updates from each of the devices and updates the global model. And then we repeat this process over multiple rounds of training. And in reality, we don't train the model on all the devices at once. Instead, we sample just a fraction of them of those that are plugged in and idle at night.

Speaker 1 ([01:59](https://www.happyscribe.com/transcriptions/f73ad59d129749f4a93b7e653d4b4d4e/edit?onboarding=true&position=119.96&utm_source=happyscribe&utm_medium=document_deep_link&utm_campaign=editor_copy_section&utm_content=f73ad59d129749f4a93b7e653d4b4d4e))  
So you don't actually notice the model being trained on your device. You might be wondering what makes this setting different to distributed training. On GPUs, there are two main factors, communication and heterogeneity. In GPU clusters, all devices are on the same network, so communication is relatively quick. However, in federated learning, devices have to communicate over WiFi, and so this is much slower than the computation and becomes the bottleneck.

Speaker 1 ([02:28](https://www.happyscribe.com/transcriptions/f73ad59d129749f4a93b7e653d4b4d4e/edit?onboarding=true&position=148.68&utm_source=happyscribe&utm_medium=document_deep_link&utm_campaign=editor_copy_section&utm_content=f73ad59d129749f4a93b7e653d4b4d4e))  
In practice, heterogeneity comes in two forms, the devices and the data. So first, the devices. Devices vary. Some phones are faster than others, and you might be using sensor data from not just your phone, but your watch, which has much less CPU power and data also varies. Some devices have much more data than others.

Speaker 1 ([02:49](https://www.happyscribe.com/transcriptions/f73ad59d129749f4a93b7e653d4b4d4e/edit?onboarding=true&position=169.62&utm_source=happyscribe&utm_medium=document_deep_link&utm_campaign=editor_copy_section&utm_content=f73ad59d129749f4a93b7e653d4b4d4e))  
Think about how many photos you take on your camera. And unlike traditional distributed training, these data distributions can vary between devices. We say they're noniid, and this could be because one user decides to take photos of landscape primarily, while another user decides to take photos of food. The environment also plays a factor. Think of taking photos of wildlife in Australia, where you might see kangaroos, versus in London, where you definitely wouldn't see kangaroos.

Speaker 1 ([03:20](https://www.happyscribe.com/transcriptions/f73ad59d129749f4a93b7e653d4b4d4e/edit?onboarding=true&position=200.53&utm_source=happyscribe&utm_medium=document_deep_link&utm_campaign=editor_copy_section&utm_content=f73ad59d129749f4a93b7e653d4b4d4e))  
And finally, in federated learning, as we've touched on before, not all devices will be available to participate. They might run out of battery or drop the WiFi connection, and so have to drop out of training. Now let's move on to the seminal paper in federated learning. This introduces an algorithm, fed average, which tries to train a shared model across clients. It does this by trying to minimize an overall global loss, that is, a weighted average of the individual clients losses.

Speaker 1 ([03:52](https://www.happyscribe.com/transcriptions/f73ad59d129749f4a93b7e653d4b4d4e/edit?onboarding=true&position=232.01&utm_source=happyscribe&utm_medium=document_deep_link&utm_campaign=editor_copy_section&utm_content=f73ad59d129749f4a93b7e653d4b4d4e))  
So in this function, there are K clients, and each client has its own loss function, FK, which it computes on device. We then weight each of the losses by the size of the client's data set. So devices with larger data sets will have correspondingly larger weighted losses. Now onto the algorithm itself. We execute this algorithm for a number of rounds of training.

Speaker 1 ([04:18](https://www.happyscribe.com/transcriptions/f73ad59d129749f4a93b7e653d4b4d4e/edit?onboarding=true&position=258.34&utm_source=happyscribe&utm_medium=document_deep_link&utm_campaign=editor_copy_section&utm_content=f73ad59d129749f4a93b7e653d4b4d4e))  
First, we sample a fraction C of the k clients. So in this case we have that K equals six and C equals 0.5. We send the current round weights to each client K. So for the teeth round, the weight is denoted by WT. The clients then run stochastic gradient descent on their local data for EEPOCs.

Speaker 1 ([04:43](https://www.happyscribe.com/transcriptions/f73ad59d129749f4a93b7e653d4b4d4e/edit?onboarding=true&position=283.41&utm_source=happyscribe&utm_medium=document_deep_link&utm_campaign=editor_copy_section&utm_content=f73ad59d129749f4a93b7e653d4b4d4e))  
Once they've done that, they send the updated weights back to the server. And once the server has received all the weights, it aggregates them, taking the weighted average. And then we repeat for the next round, and so on. Now, Fed average works well in practice, but it isn't perfect. It makes a number of simplifying assumptions.

Speaker 1 ([05:04](https://www.happyscribe.com/transcriptions/f73ad59d129749f4a93b7e653d4b4d4e/edit?onboarding=true&position=304.73&utm_source=happyscribe&utm_medium=document_deep_link&utm_campaign=editor_copy_section&utm_content=f73ad59d129749f4a93b7e653d4b4d4e))  
For one, it assumes that all sample devices will complete E epochs of local stochastic gradient descent. But some devices take longer than others, and these stragglers can actually hurt the speed of convergence. So in practice, fed average just drops the straggler. But what happens if 90% of your devices are stragglers? And secondly, it's not guaranteed to converge when our data is highly heterogeneous.

Speaker 1 ([05:31](https://www.happyscribe.com/transcriptions/f73ad59d129749f4a93b7e653d4b4d4e/edit?onboarding=true&position=331.67&utm_source=happyscribe&utm_medium=document_deep_link&utm_campaign=editor_copy_section&utm_content=f73ad59d129749f4a93b7e653d4b4d4e))  
Fed average weights devices by the proportion of the data that they own, and so it might favor certain devices performances at the expense of others. Therefore, improvements to Fed average have been proposed in the last few years, the first Fed prox allows devices to do variable amounts of work. Now, naively, you might think that this favors devices that can run more sets of gradient sent in the same amount of time and therefore change the weights of their models much more. So Pedprox introduces a regularization term, or proximal term, that penalizes large changes in weight. This also helps convergence on highly heterogeneous data, as you can think of this proximal term as penalizing the model from changing too much on one single device, and we control the amount that's penalized by this hyperameter Mu.

Speaker 1 ([06:28](https://www.happyscribe.com/transcriptions/f73ad59d129749f4a93b7e653d4b4d4e/edit?onboarding=true&position=388.25&utm_source=happyscribe&utm_medium=document_deep_link&utm_campaign=editor_copy_section&utm_content=f73ad59d129749f4a93b7e653d4b4d4e))  
And with heterogeneous data, model performance can vary a lot as different distributions require different sets of features like food versus landscapes. So there are two approaches to fixing this. The first QFED average seeks to make the shared model learnt a lot more fair. That is, it performs similarly on all devices. So rather than weighting devices by the proportion of data that they have, we penalize worse performing devices more, incentivizing the model to improve performance on these devices.

Speaker 1 ([07:00](https://www.happyscribe.com/transcriptions/f73ad59d129749f4a93b7e653d4b4d4e/edit?onboarding=true&position=420.23&utm_source=happyscribe&utm_medium=document_deep_link&utm_campaign=editor_copy_section&utm_content=f73ad59d129749f4a93b7e653d4b4d4e))  
And you can see this as we raise the loss to the power of Q Plus one, we can tune this hyperparameter Q. So the larger Q is, the more these worst performing clients dominate the overall loss, and so the more fair it becomes. Another approach per set average seeks to train a model that can be personalized to each device after running a few sets of local gradient descent, and so the loss function now changes from the loss on the current weights at that round to the weights after a step of radiant ascent. If you're familiar with meta learning, this uses the mammal approach and formulates federated learning as a multitask problem where each client's distribution is a separate task. As part of my master's degree at the University of Cambridge, I've actually compared perfed average against QFED average on heterogeneous data, so be sure to hit the like button and comment below if you'd like to see a video on that federated learning as a field has seen an explosion in the number of papers submitted to archive year on year, and it can be overwhelming to search through them yourself.

Speaker 1 ([08:59](https://www.happyscribe.com/transcriptions/f73ad59d129749f4a93b7e653d4b4d4e/edit?onboarding=true&position=539.29&utm_source=happyscribe&utm_medium=document_deep_link&utm_campaign=editor_copy_section&utm_content=f73ad59d129749f4a93b7e653d4b4d4e))  
But what if we wanted to implement Federated learning ourselves? Luckily for us, there are a few frameworks that can help us out. For Tensorflow we have Tensorflow, Federated, and for Pytorch we have PySift, developed by the Openmind community. Both of these frameworks have vibrant communities backing them that are growing. These frameworks integrate tightly with their respective deep learning frameworks, but they're relatively low level.

Speaker 1 ([09:24](https://www.happyscribe.com/transcriptions/f73ad59d129749f4a93b7e653d4b4d4e/edit?onboarding=true&position=564.87&utm_source=happyscribe&utm_medium=document_deep_link&utm_campaign=editor_copy_section&utm_content=f73ad59d129749f4a93b7e653d4b4d4e))  
If you're not using Fedaverage, then you're required to implement the Federated Learning strategy yourself. There's a third federated learning framework spun out of the University of Cambridge called flower that takes a different approach. Instead of being tied to a particular framework, it's agnostic and you can plug and play different components. For example, within the library you have an option to swap out fed average for a different strategy, such as QFED average. And although it has a small community, it's growing very rapidly, so it's one to look out for.