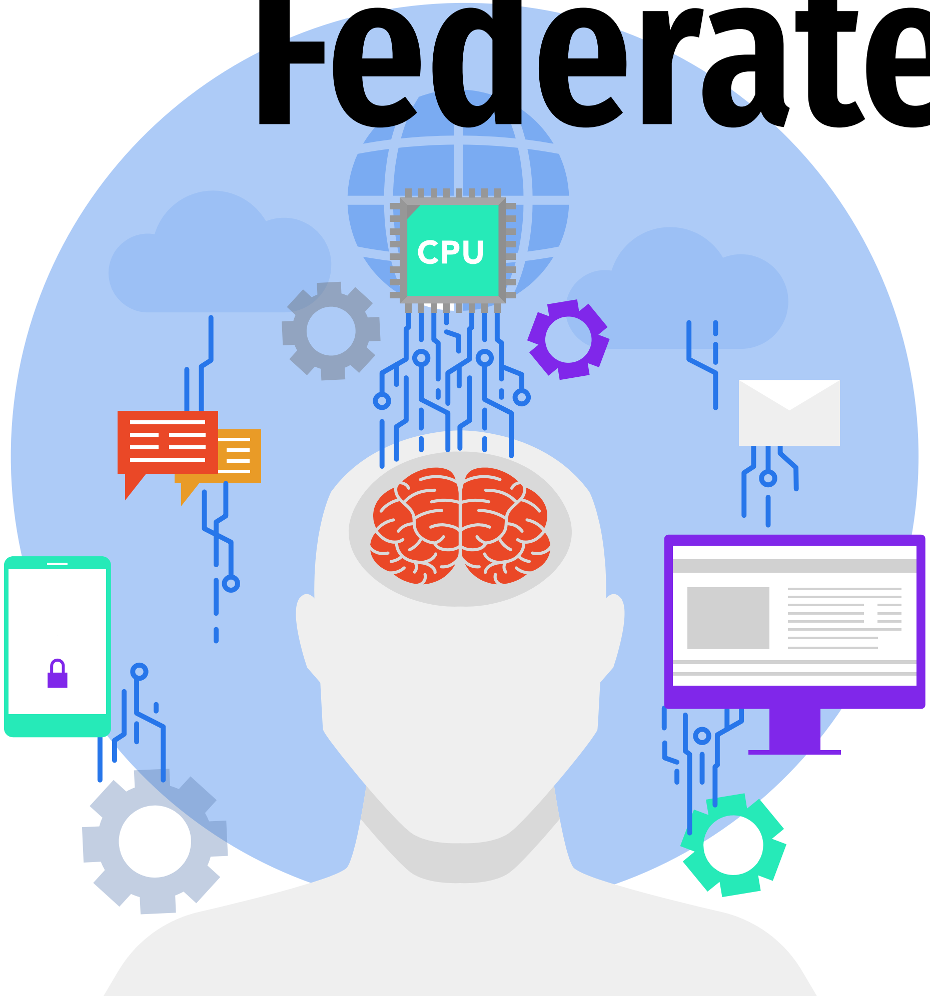
Federated Learning



Communication-Efficient Learning of Deep Networks from Decentralized Data

H. Brendan McMahan,
Eider Moore,
Daniel Ramage,
Seth Hampson,
Blaise Aguera y Arcas

Today's Schedule

Federated learning tutorial

01

Why

What problem does F.L. solve?

02

What

What does F.L. involve?

03

How

How did F.L. start?

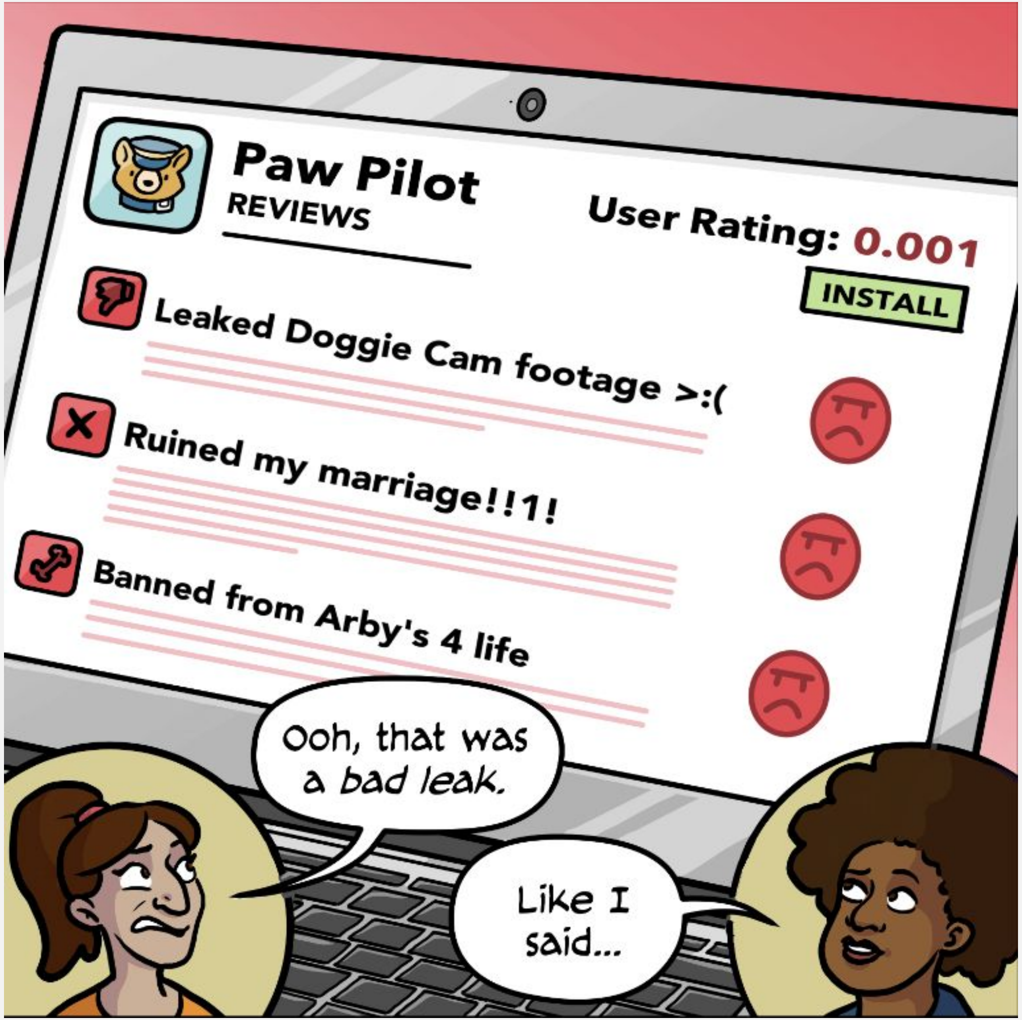
04

Future

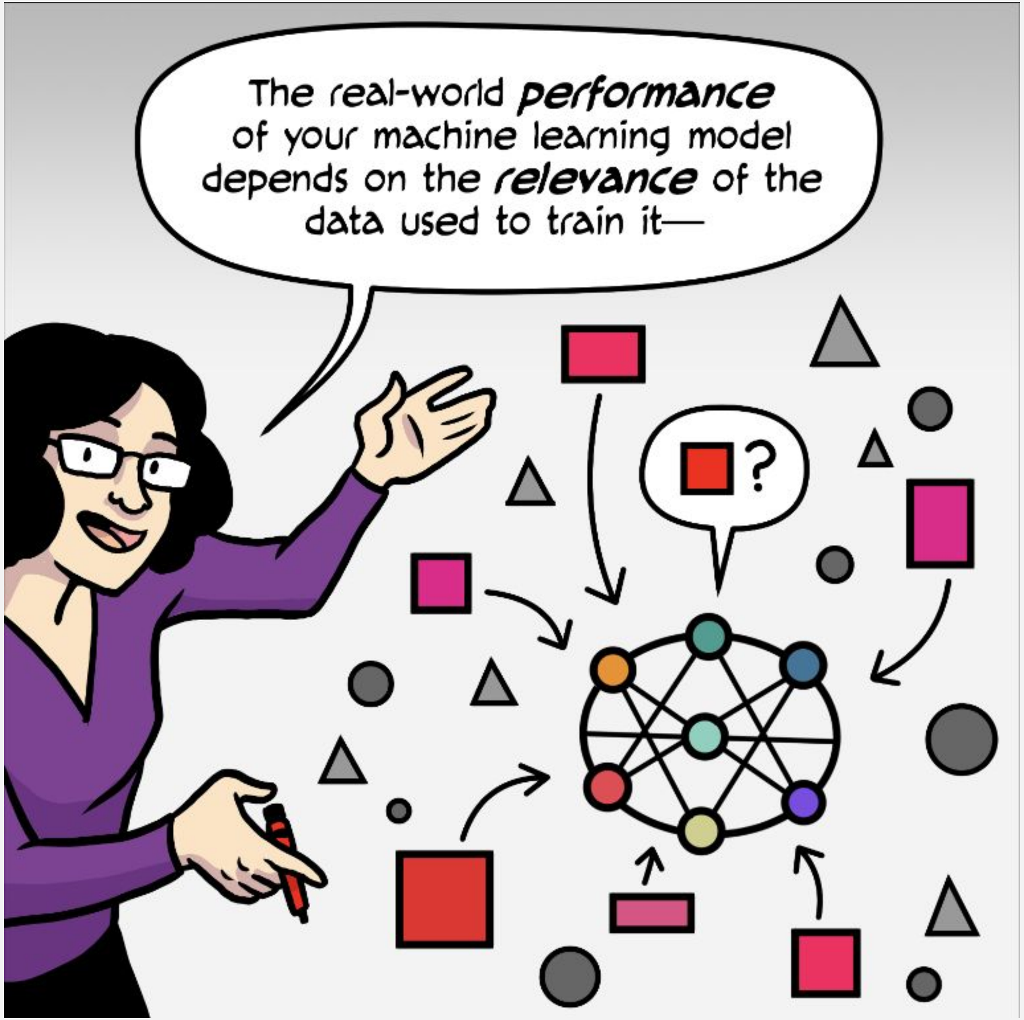
What were the future directions of F.L.?

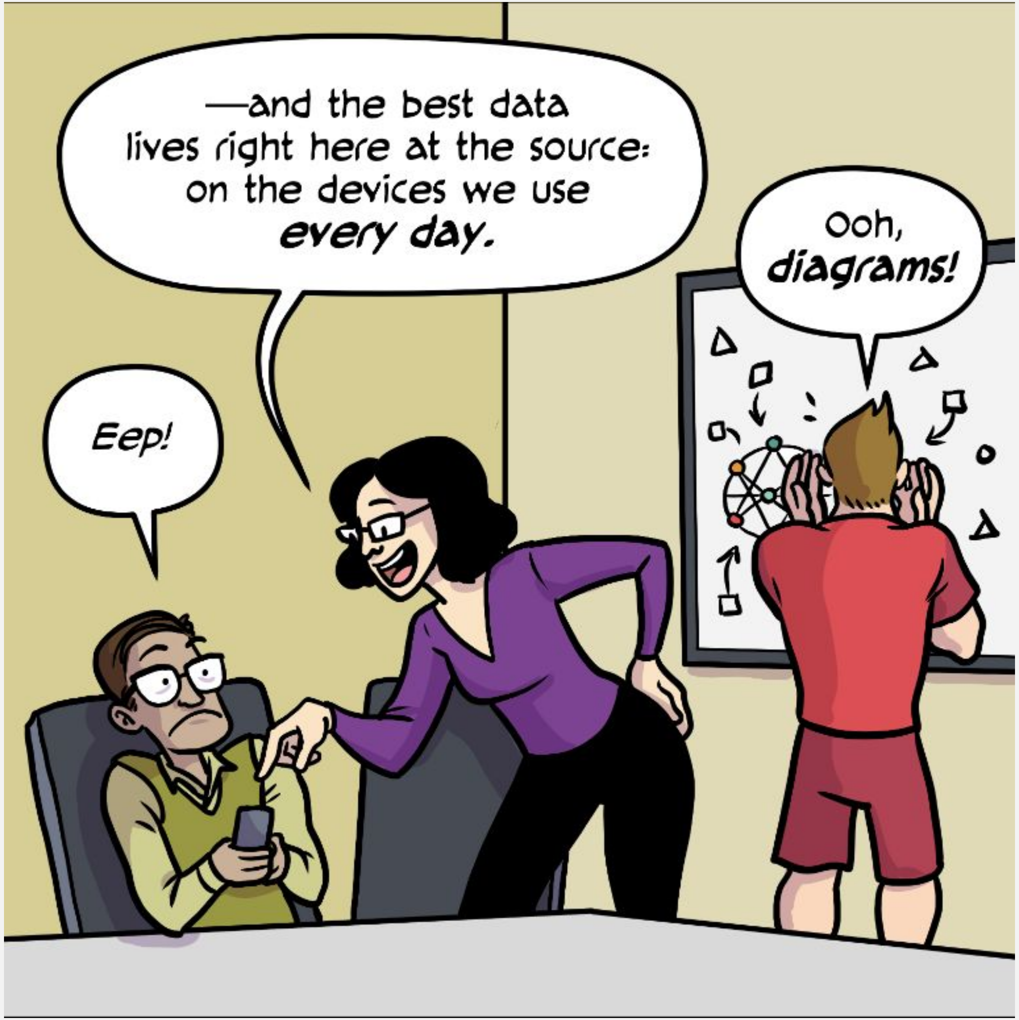
What problem does F.L. solve?


It seemed like a good idea at the time...
PAW PILOT
YOUR CONSTANT CANINE COMPANION!
PLOT YOUR PERFECT PET PATH WITH SUGGESTED ROUTES!
FEWER VET VISITS WITH HEALTH TRACKING!
EARN CASH FOR CHOW WITH ROVER REWARDS!
ALWAYS WATCHING! ALWAYS LISTENING!
Download now!


Paw Pilot
REVIEWS
User Rating: 0.001
INSTALL
Leaked Doggie Cam footage >:(
Ruined my marriage!!1!
Banned from Arby's 4 life
Ooh, that was a *bad leak.*
Like I said...


Guardian
Royal Corgi Hostage Debacle
Parliament Evacuated
...privacy nightmare.
OUR DOGS OUR DATA
TIMES
Pooch Pilfered by Pirates


The real-world *performance* of your machine learning model depends on the *relevance* of the data used to train it—
?

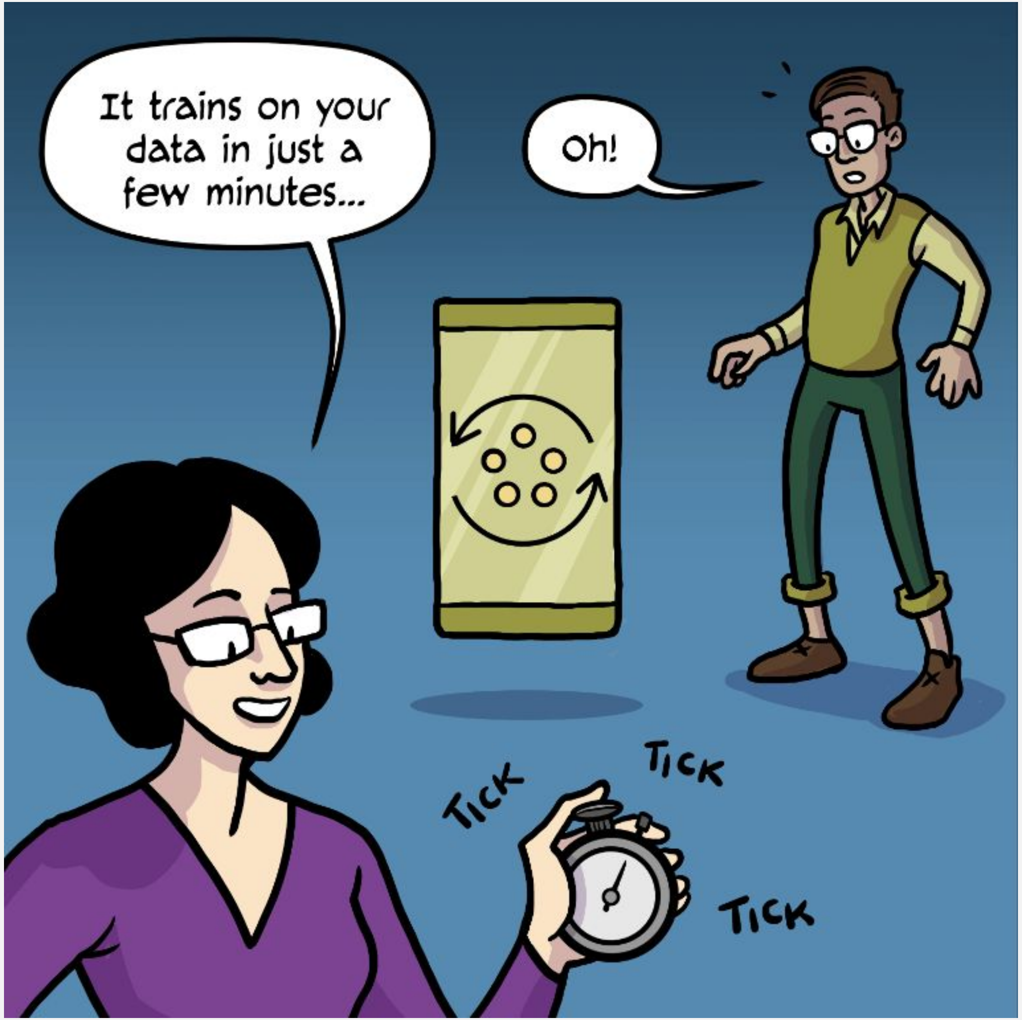

—and the best data lives right here at the source: on the devices we use *every day.*
Ooh, *diagrams!*
Eep!

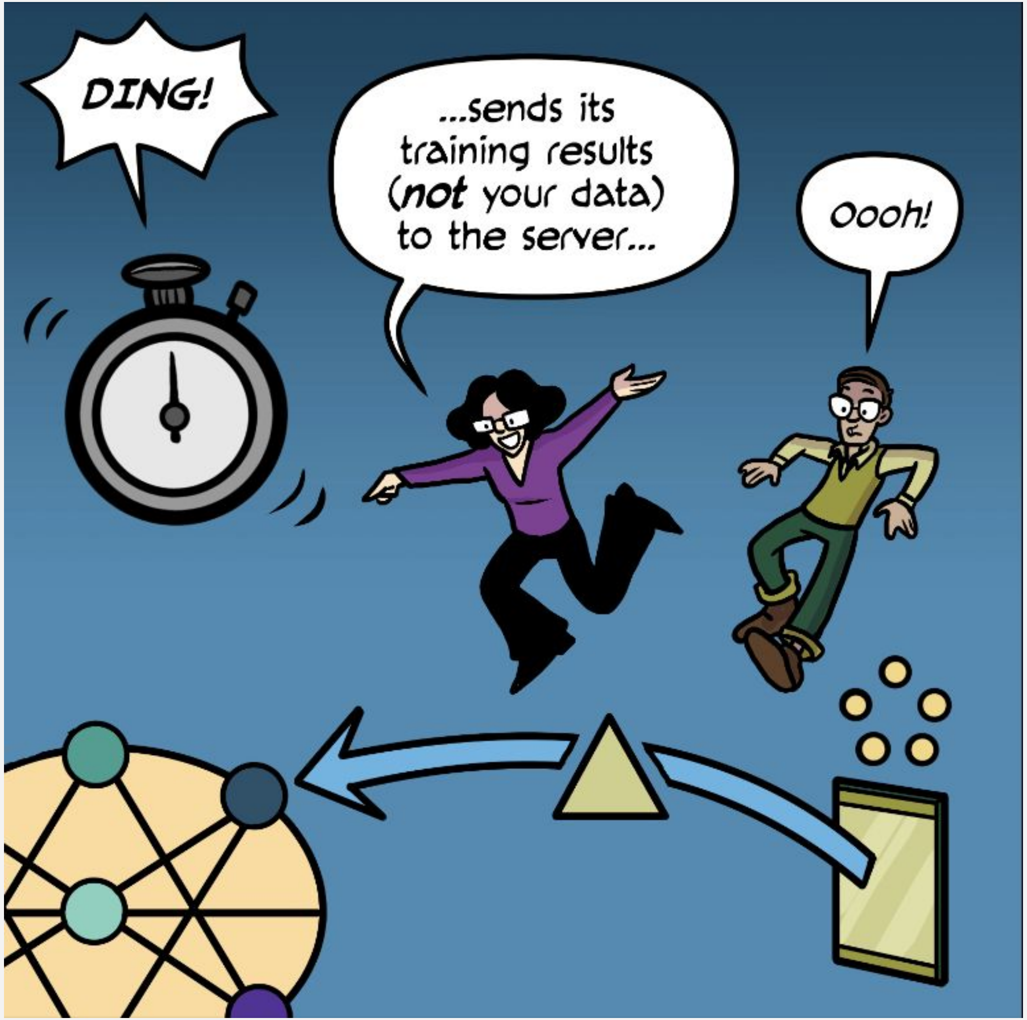

On-device data can be used to train a smarter central model and improve our users' experience.
But since there's no way we'd wanna bring that data to the server...


...the training can be brought *to the device!*


oh!
CHARGING
ON WI-FI
IDLE
We're not impacting our users or their phones at all!

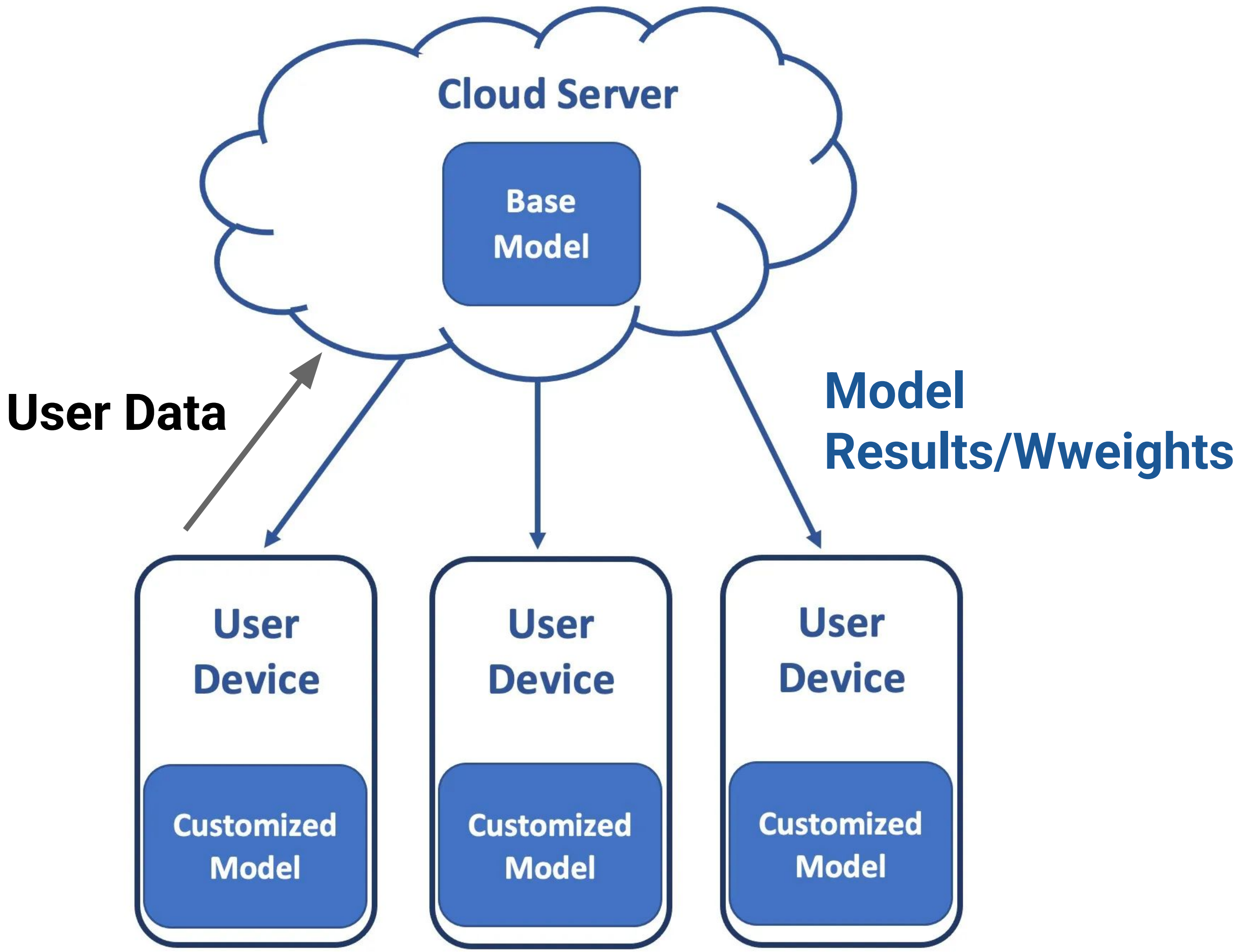

—**and so**, a subset of devices are selected—
Gyeeah!
—to receive a ***training model.***

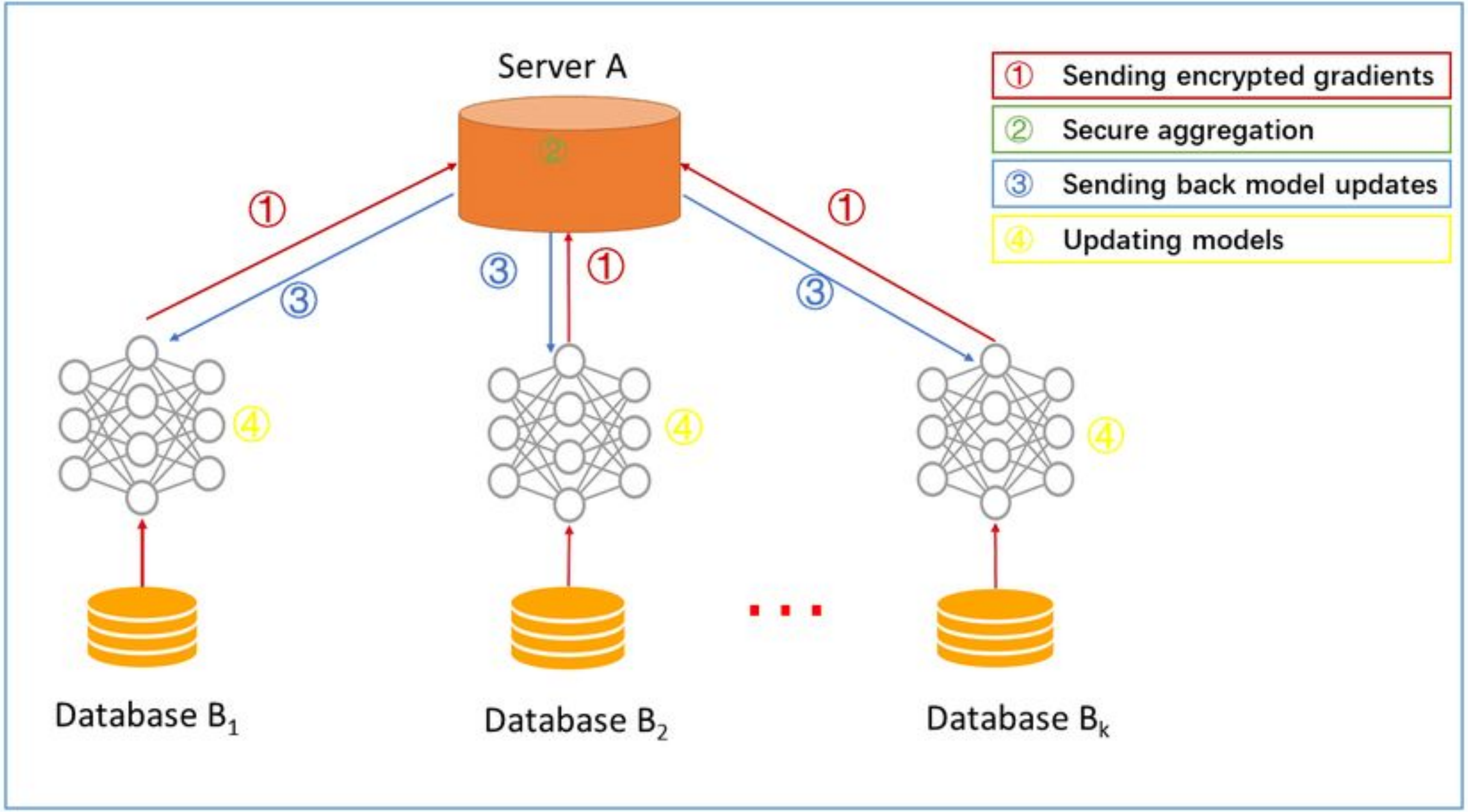

It trains on your data in just a few minutes...
Oh!
TICK
TICK
TICK


DING!
...sends its training results (*not* your data) to the server...
Oooh!


...federated learning!

Recap


Cloud Server
Base Model
User Data
Model Results/Wweights
User Device
Customized Model
User Device
Customized Model
User Device
Customized Model


Server A
① Sending encrypted gradients
② Secure aggregation
③ Sending back model updates
④ Updating models
Database B_1
Database B_2
Database B_k

The pros, cons, and the algorithm

Federated Learning works best when:

Federated Learning

Real Data Required

Training on “real world data” provides a distinct advantage over training on proxy data that is generally available in the data center

Private or Large Data

This data is privacy sensitive or large in size (compared to the size of the model), so it is preferable not to log it to the data center purely for the purpose of model training

Simplified Labeling

For supervised tasks, labels on the data can be inferred naturally from user interaction

Challenges of federated Optimization

- **Non-IID**: The training data is highly user specific, hence any particular user's local dataset will not be representative of the population distribution.
- **Unbalanced**: User app usage rates are not uniform.
- **Massively Distributed**: the number of clients participating in an optimization to be much larger than the average number of examples per client.
- **Limited Communication**: Phones are usually offline and can't communicate with the server.
- **Communication costs:** (next slide)

Federated Learning changes your cost structure

- In data center optimization, communication costs are low, computational costs are high. So we have to use powerful GPUs to mitigate this.
- In federated learning, communication costs are high, computational costs are (relatively) low:
  - Individual datasets are small, and phones have the computational power to train on them (some even have GPUs now) so computation isn’t too big of an issue.
  - On the other hand, we will typically be limited by an upload bandwidth of 1 MB/s or less
  - Clients will typically only volunteer to participate in the optimization when they are charged, plugged-in, and on an unmetered wi-fi connection (only a few rounds per day).
  - So, the authors:
    1. Maximize the number of clients working independently (small boost)
    2. Increase computation per round per client: ie, not just 1 gradient descent (big help)

Google developed federated averaging

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
  initialize w_0
  for each round $t = 1, 2, \dots$ **do**
    $m \leftarrow \max(C \cdot K, 1)$
    $S_t \leftarrow$ (random set of m clients)
    for each client $k \in S_t$ **in parallel do**
      $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
    $m_t \leftarrow \sum_{k \in S_t} n_k$
    $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$ *// Erratum*[4]

ClientUpdate(k, w)**:** *// Run on client k*
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
  for each local epoch i from 1 to E **do**
    for batch $b \in \mathcal{B}$ **do**
      $w \leftarrow w - \eta \nabla \ell(w; b)$
  return w to server

Can add things other than SGD here if needed

Loss aggregation

Normal ML:

$$\min_{w \in \mathbb{R}^d} f(w) \qquad \text{where} \qquad f(w) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(w).$$

Num Clients

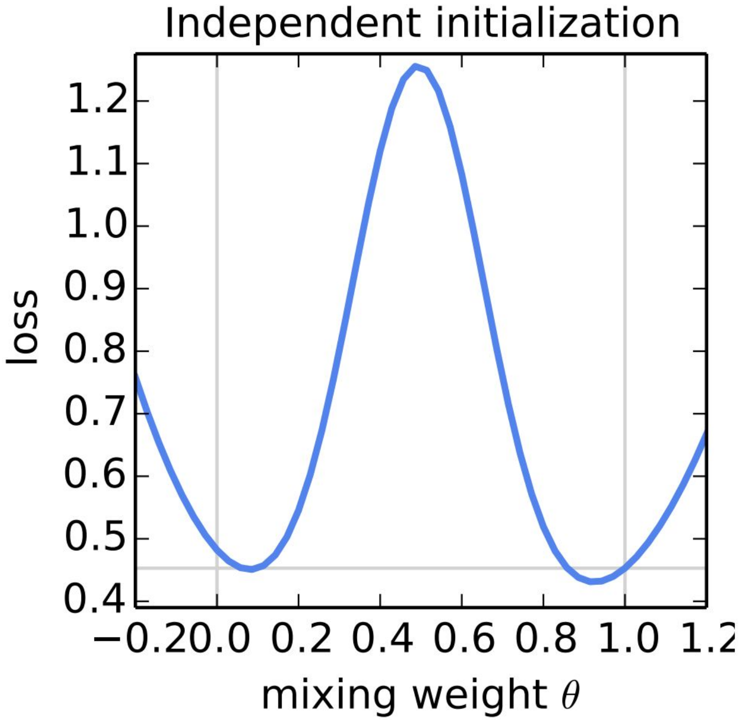
F.L. $$f(w) = \sum_{k=1}^{\boxed{K}} \boxed{\frac{n_k}{n}} \boxed{F_k(w)} \quad \text{where} \quad F_k(w) = \frac{1}{n_k} \sum_{i \in \mathcal{P}_k} f_i(w)$$

Weighted by the size of Client k's dataset

Client Loss Func.

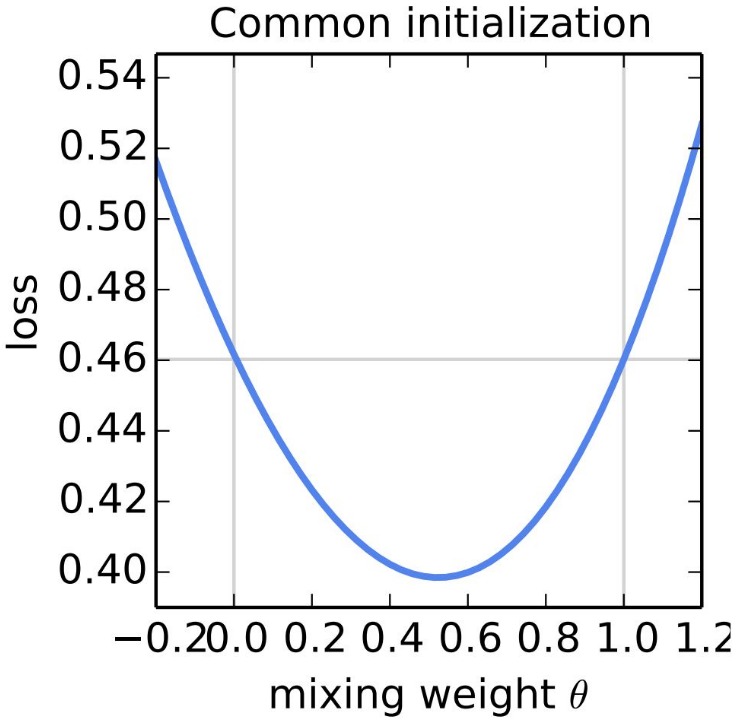
Independent vs Common Initialization

- For general non-convex objectives, averaging models in parameter space could produce an arbitrarily bad model.
- This is because the averaging process may mix models that are trapped in different local minima, leading to a suboptimal or even arbitrarily bad solution in terms of predictive performance.



Independent vs Common Initialization

- When we start two models from the same random initialization and then train each independently on a different subset of the data, we find that naive parameter averaging works surprisingly well (shown right)
- Federated Average therefore uses this shared weight initialization.



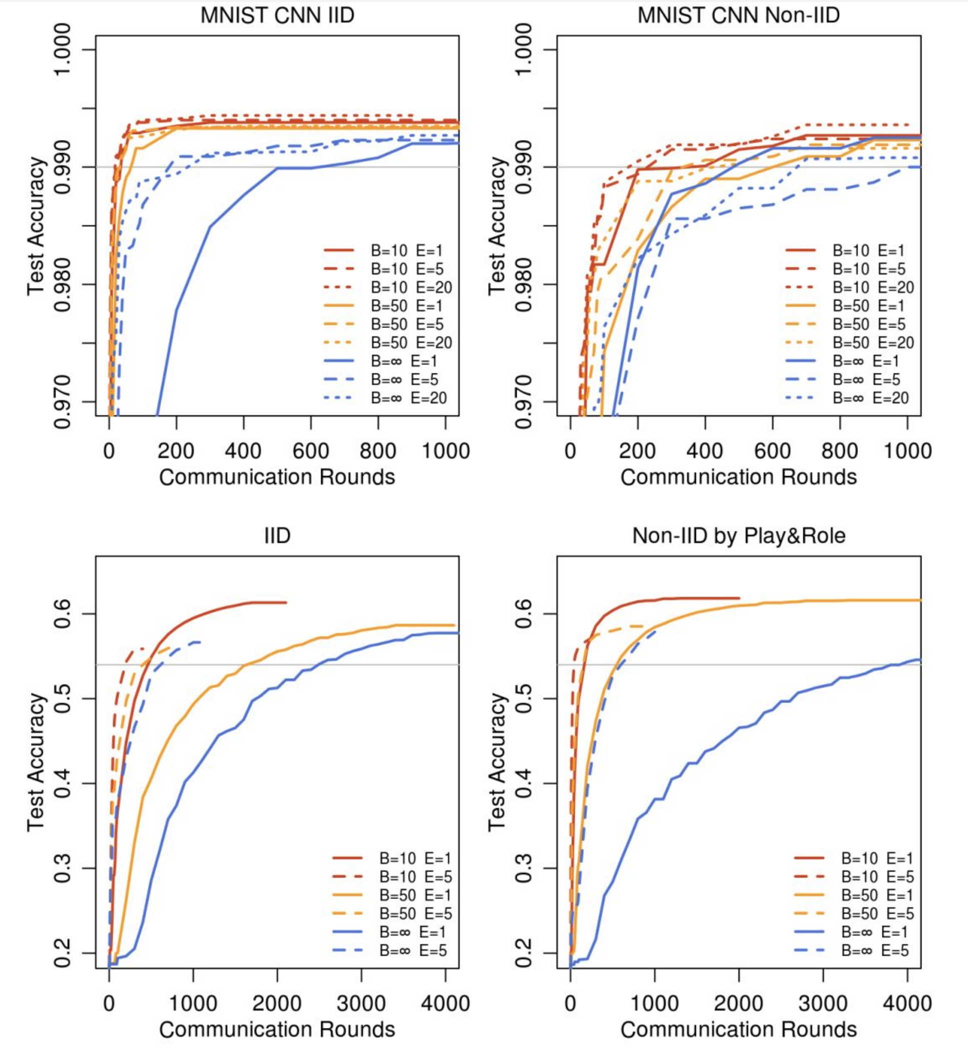
Experiment & Results

Experiment

Tests were conducted on two tasks:

- **CIFAR-10 classification task**. 3 model architectures tested over 2 data distributions:
  - **IID**: data is shuffled, and then partitioned into 100 clients each receiving 600 examples
  - **Non IID**: sort the data by digit label, divide it into 200 shards of size 300, and assign each of 100 clients 2 shards

- **Large language modeling task**:
  - Dataset is from *The Complete Works of William Shakespeare*
  - **Non-IID Partition:**
    - Construct a client dataset for every speaking role with at least 2 lines (1146)
    - 80/20 train/test split resulting in: 3,564,579 chars in the training set, and 870,014 chars in the test set
    - Roles are substantially unbalanced, with some roles at the 2 line limit and many with much higher contributions (ie. romeo)
  - **IID Partition:** balanced and IID version of the dataset, also with 1146 clients.
- **Large Language task 2**: LLM(LSTM) on social network posts 5000 words limit for 500,000 clients


MNIST CNN IID
MNIST CNN Non-IID
IID
Non-IID by Play&Role
Test Accuracy
Communication Rounds
B=10 E=1
B=10 E=5
B=10 E=20
B=50 E=1
B=50 E=5
B=50 E=20
B=∞ E=1
B=∞ E=5
B=∞ E=20

Results

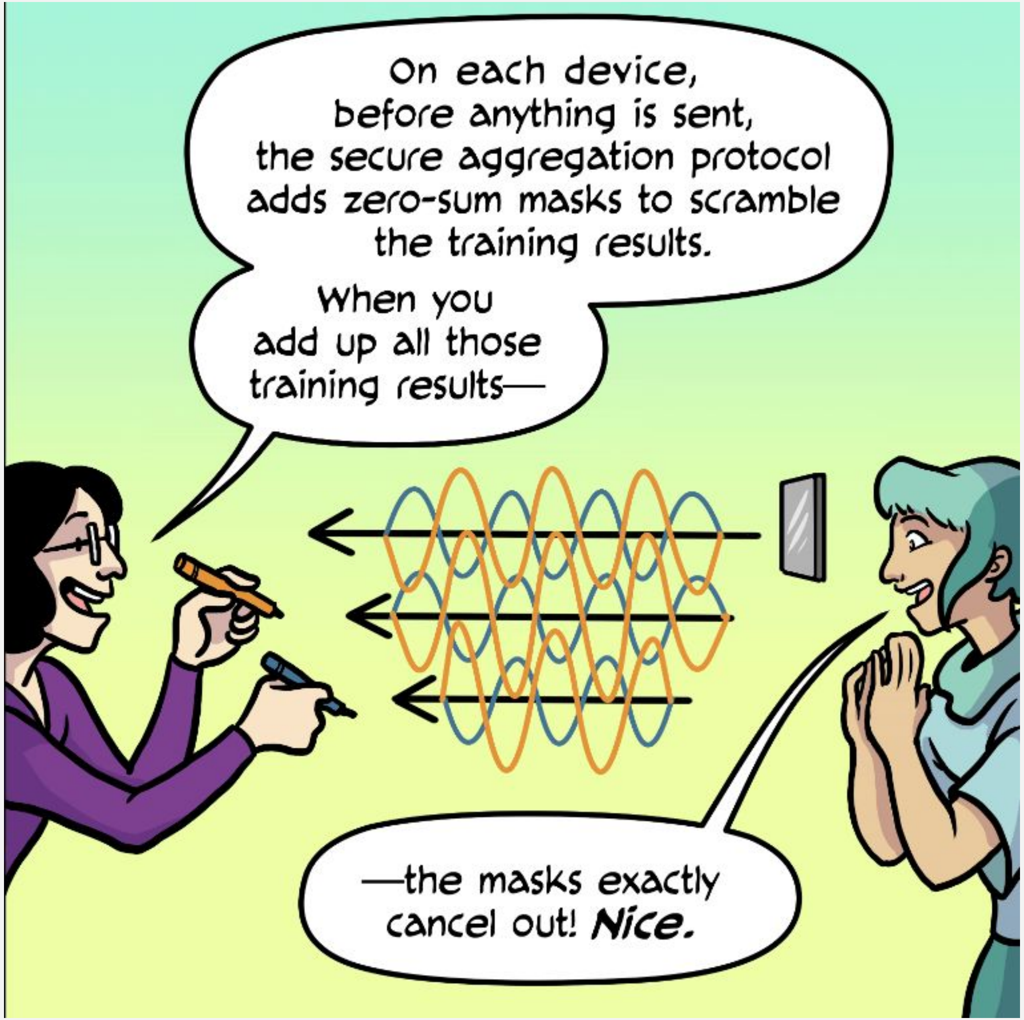
- One round of averaging per client is enough for a specific dataset instance: “we would expect that while one round of averaging might produce a reasonable model, additional rounds of communication (and averaging) would not produce further improvements.”
- Increasing the number of computations per client will dramatically decrease the computation costs required to get to model convergence.
    - For MNIST: the authors report a 35-46 times decrease for IID data, and 2.8-3.7x decrease for Non-IID data.
    - For Shakespeare the authors report a 95x speedup for Non-IID data and a 13x speedup for IID data, likely due to the fact that some roles have relatively large local datasets, which makes increased local training more valuable.
- FedAvg works better across all tests than FedSGD.
- The author’s report that there is an optimal number of clients chosen per batch update, where increasing past that number did not help model convergence. They optimum value was 10%.

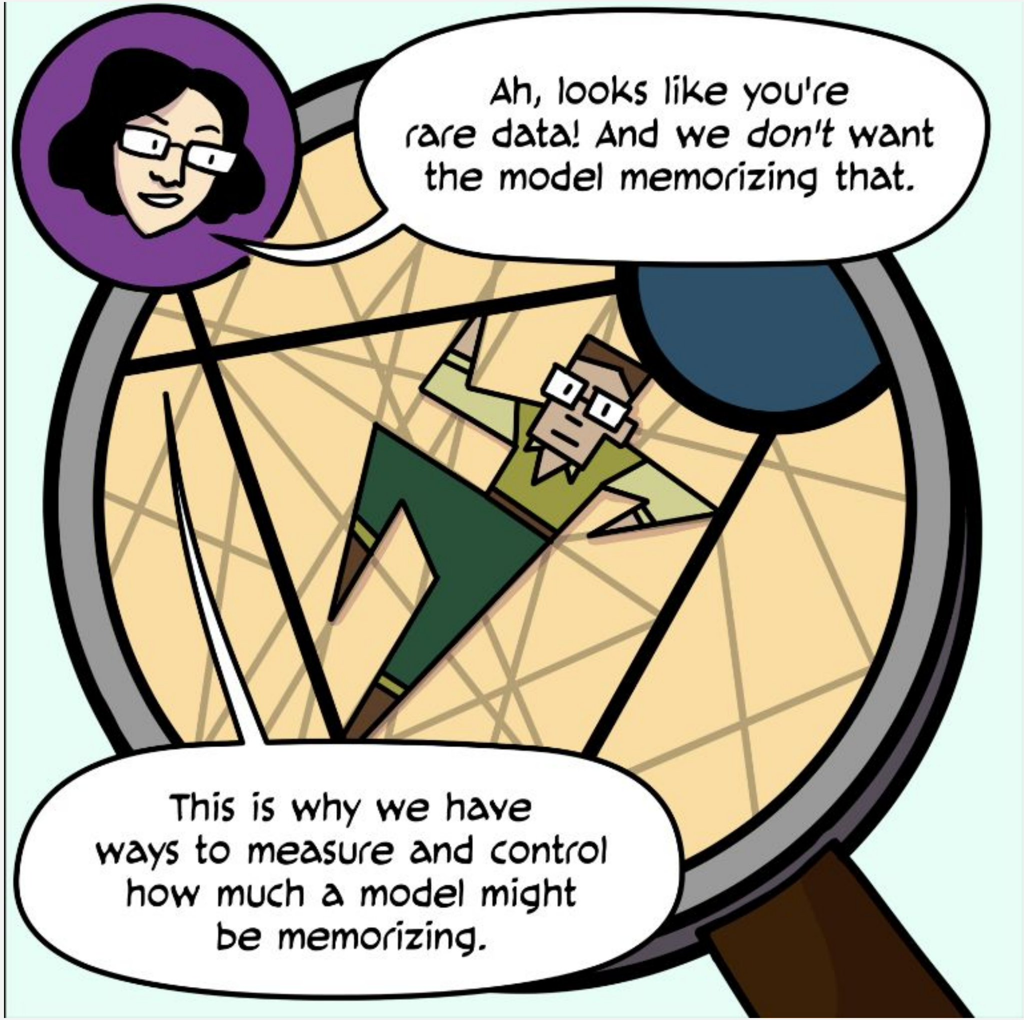
Weaknesses of FedAvg (unaddressed in paper)

- Assumes that all devices compute E epochs per round, but some devices will lose eligibility mid-way through training or perform much slower than the average. So these stragglers hurt the convergence rate (drop them).

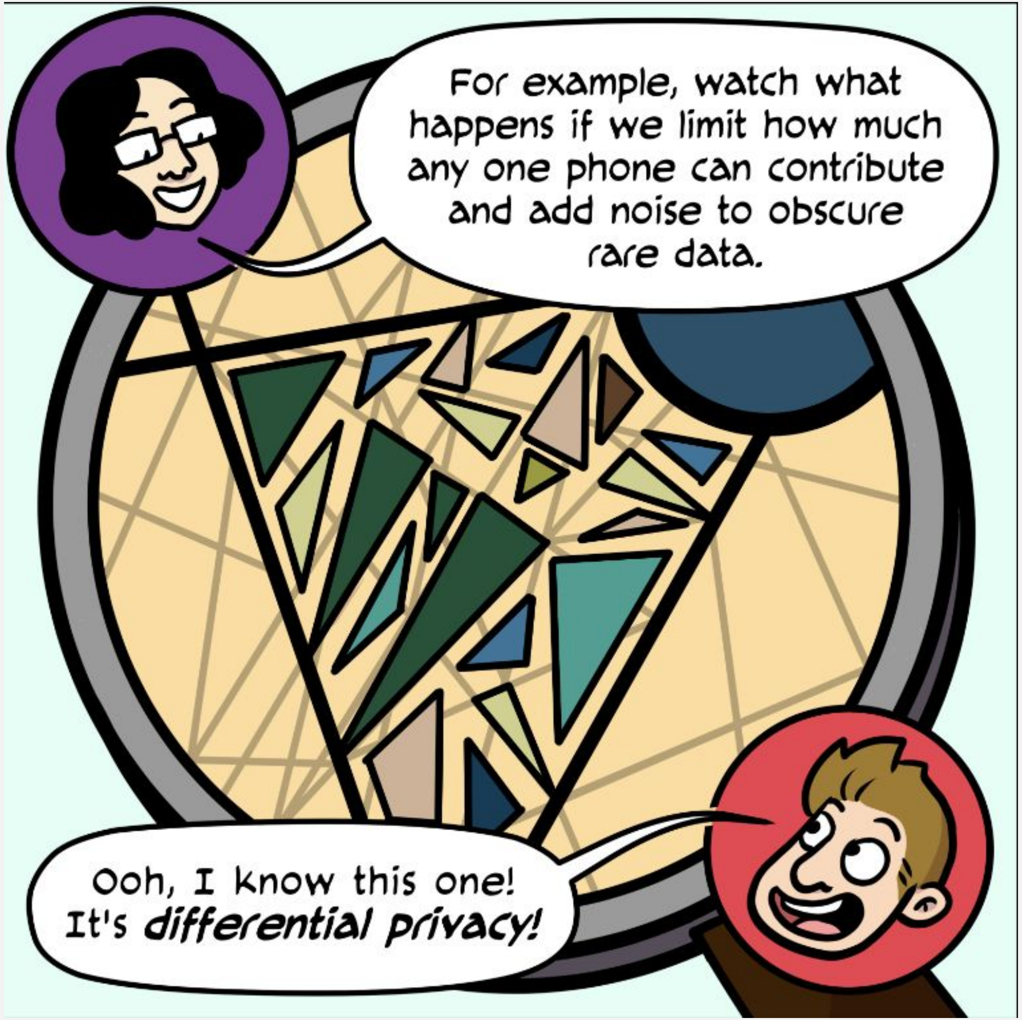
- Baseline FedAvg weights users based on the size of their dataset, which isn’t always ideal.

Future Work


On each device, before anything is sent, the secure aggregation protocol adds zero-sum masks to scramble the training results.
When you add up all those training results—
—the masks exactly cancel out! *Nice.*



Ah, looks like you're rare data! And we *don't* want the model memorizing that.
This is why we have ways to measure and control how much a model might be memorizing.


For example, watch what happens if we limit how much any one phone can contribute and add noise to obscure rare data.
Ooh, I know this one! It's *differential privacy!*

Differential Policy & Secure Multi-Party Computation

- Private information can still be captured if the transmission of model weights is stolen, so we need to use a differential policy in communication to ensure that the output of a computation or query does not reveal information about any individual data point in the dataset. We do this by adding a controlled amount of noise that we can then remove in the server.
- We also need security multi-party computation to keep the data private within a client batch.

Thank You!

Questions?

Extra Slides

Why is it called Horizontal Federated Learning?

- In horizontal federated learning, different data owners (such as multiple devices or organizations) share a common feature space. This means that the data sources have the same types of features but different data points.
    - Example: learning text completion based on many user’s texting habits

- In vertical federated learning, the data sources have different sets of features, but there is some common overlap between them. This means that each party has data on different attributes or dimensions, and they are interested in learning from the intersection of these attributes.
    - Example: You want to predict medical health of a patient, so you learn their bone health from one hospital, impact of smoking habits from another clinic etc.