Federated

A new interesting approach

Guess what we are talking about today ...

of course...

Machine Learning!

State of the art

There are mainly three principles machine learning *paradigms*.

State of the art

Supervised Learning

Starting from a set of examples, the algorithm can learn some hidden pattern in the data. For each example there is the output. In this way the algorithm can learn from a “knowledge” to predict new data.

Unsupervised Learning

Starting from a set of examples, the algorithm try learn some hidden pattern in the data. The dataset doesn’t provide solutions for examples.

I.e. clustering.

Reinforcement Learning

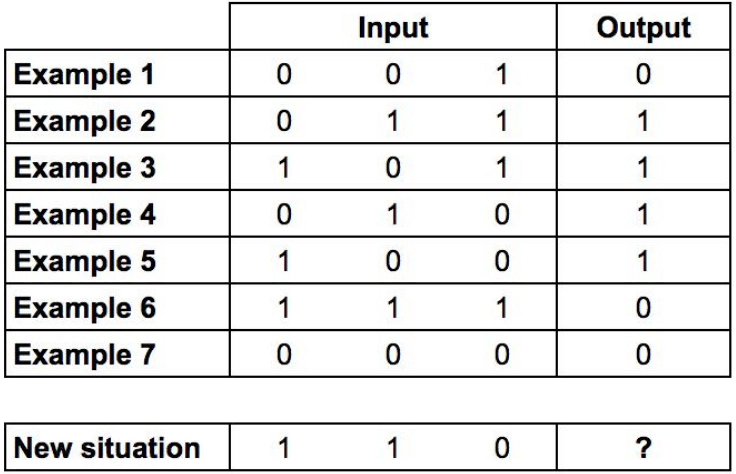
Briefly, the algorithm learns from its mistakes and expands the ways to reach the goal.

Applications: chess, AlphaGo, ...

We focus on supervised learning

Supervised Learning

- We have a dataset and we split it in *training set* and *test set* (we explain why we do this later)
- How the training set is:

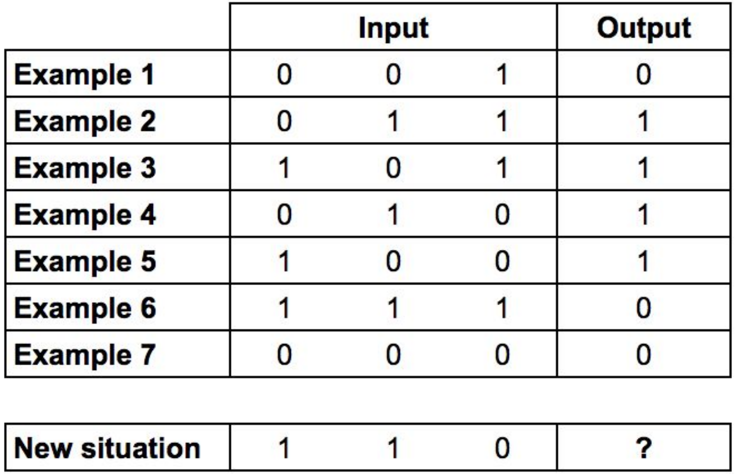


	Input			Output
Example 1	0	0	1	0
Example 2	0	1	1	1
Example 3	1	0	1	1
Example 4	0	1	0	1
Example 5	1	0	0	1
Example 6	1	1	1	0
Example 7	0	0	0	0

New situation	1	1	0	**?**

Supervised Learning

- The algorithm learns from the training set the pattern inside the data
- Through the test set, we test our algorithm if it is able to predict new data (without providing the output) with a good likelihood.



	Input			Output
Example 1	0	0	1	0
Example 2	0	1	1	1
Example 3	1	0	1	1
Example 4	0	1	0	1
Example 5	1	0	0	1
Example 6	1	1	1	0
Example 7	0	0	0	0

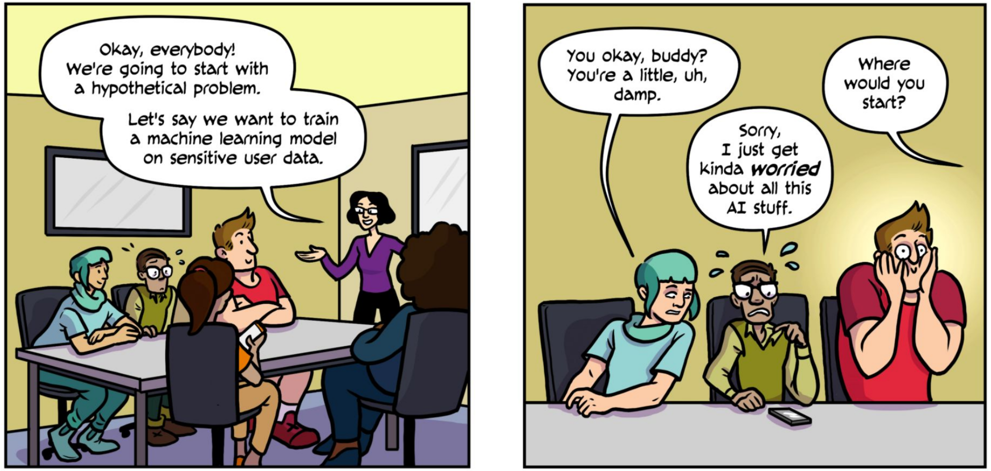
New situation	1	1	0	?

How to collect data?

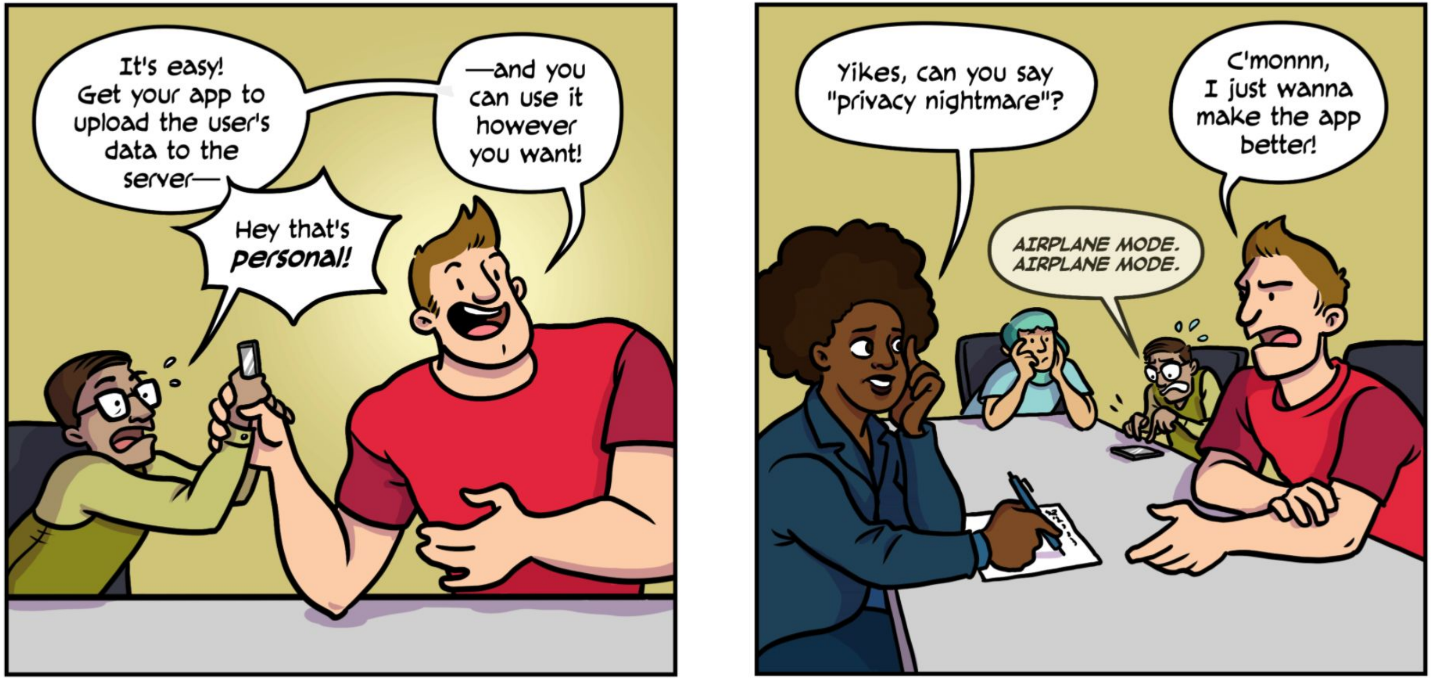
How to collect data?

Nowadays, the standard pipeline for performing supervised learning is to start with data collection. **A huge amount of data**.

Source of the comic: https://federated.withgoogle.com/

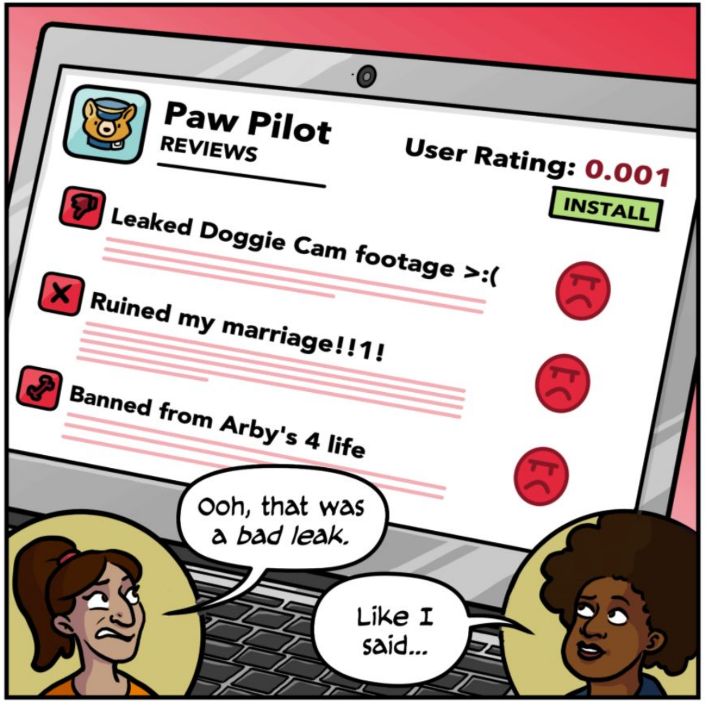

Okay, everybody! We're going to start with a hypothetical problem.
Let's say we want to train a machine learning model on sensitive user data.
You okay, buddy? You're a little, uh, damp.
Sorry, I just get kinda *worried* about all this AI stuff.
Where would you start?

How to collect data?

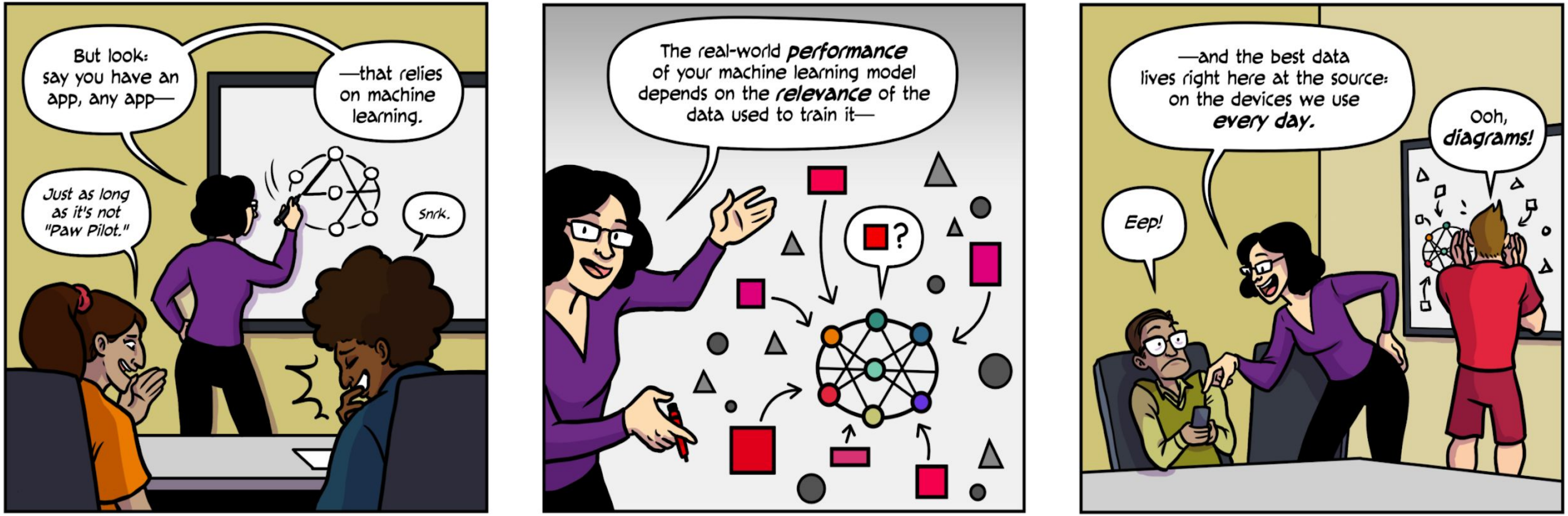

It's easy! Get your app to upload the user's data to the server—
—and you can use it however you want!
Hey that's *personal!*
Yikes, can you say "privacy nightmare"?
AIRPLANE MODE. AIRPLANE MODE.
C'monnn, I just wanna make the app better!

How to collect data?

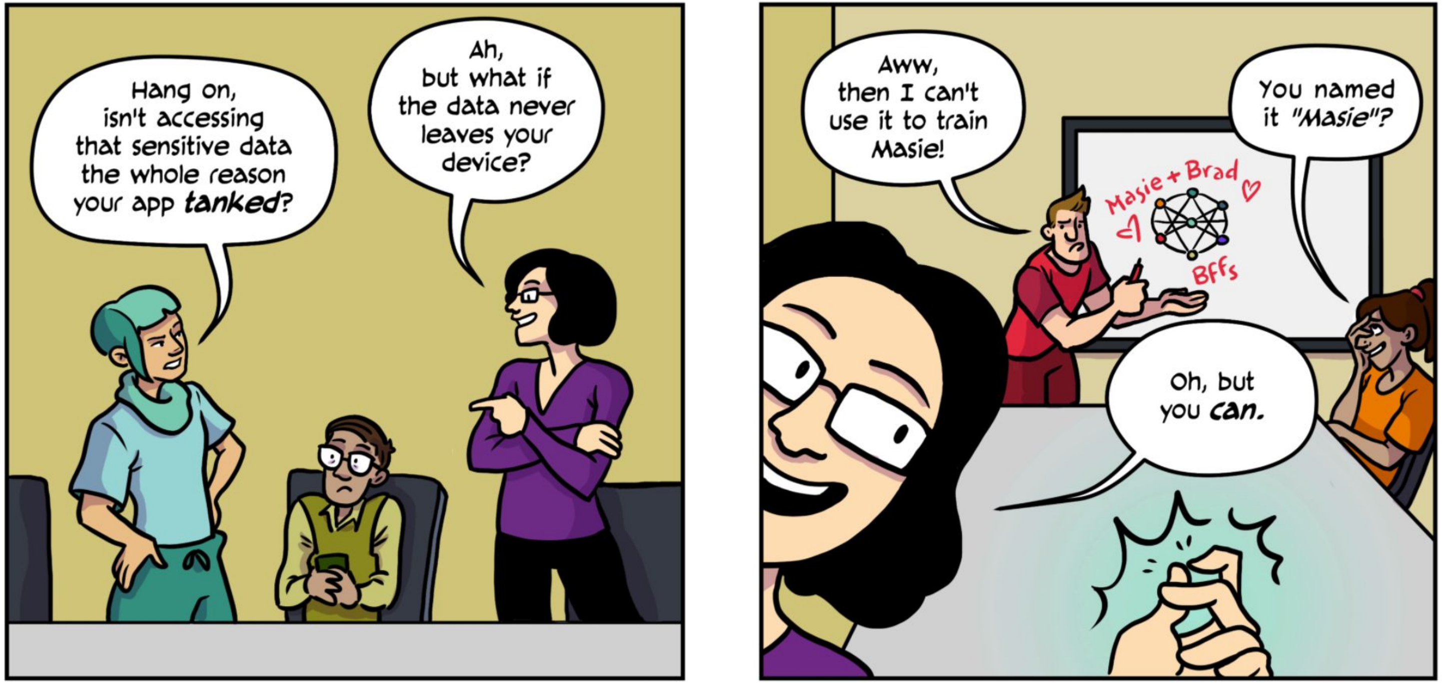
It seemed like a good idea at the time...
PAW PILOT
YOUR CONSTANT CANINE COMPANION!
PLOT YOUR PERFECT PET PATH WITH SUGGESTED ROUTES!
FEWER VET VISITS WITH HEALTH TRACKING!
EARN CASH FOR CHOW WITH ROVER REWARDS!
ALWAYS WATCHING! ALWAYS LISTENING!
Download now!


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...

How to collect data?


But look: say you have an app, any app—
—that relies on machine learning.
Just as long as it's not "Paw Pilot."
Snrk.
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.*
Eep!
Ooh, *diagrams!*

How to collect data? A new approach


Hang on, isn't accessing that sensitive data the whole reason your app *tanked*?
Ah, but what if the data never leaves your device?
Aww, then I can't use it to train Masie!
Masie + Brad
BFFs
You named it *"Masie"*?
Oh, but you *can*.

A new approach: Federated Learning

Welcome to the world of *federated learning!*
A-buahhh?
We're about to train a *centralized* model on *decentralized* data.

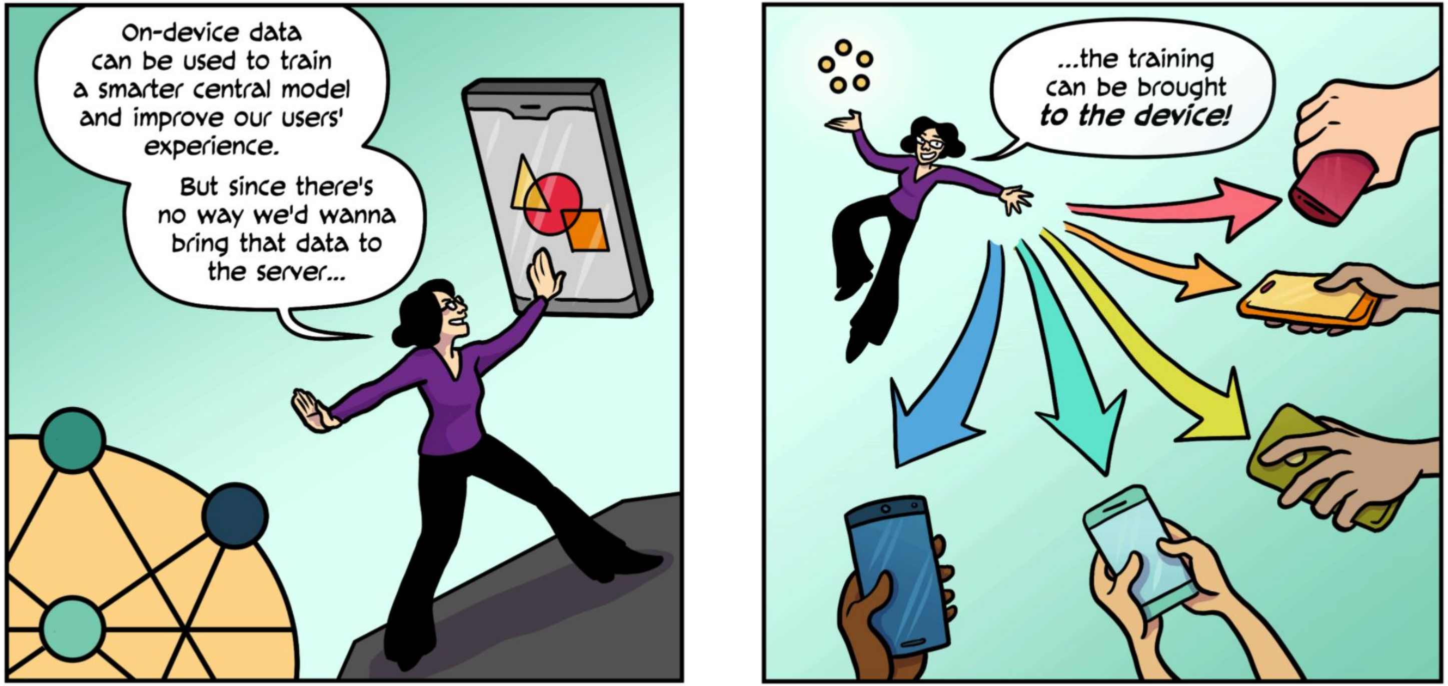
A new approach: Federated Learning

With the standard pipeline, the machine learning process trains the model on **centralized** data.

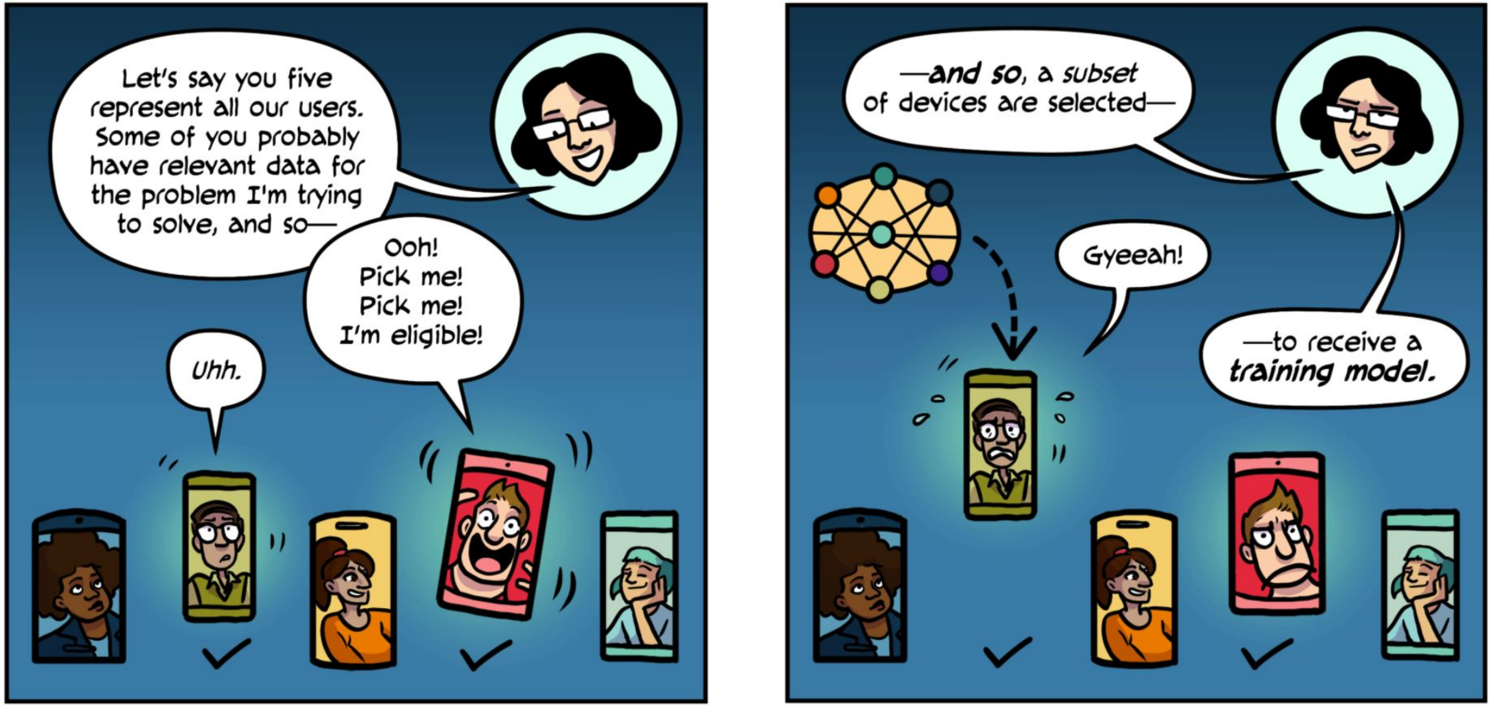
But if we train our model on **decentralized** data?

In practice, how?

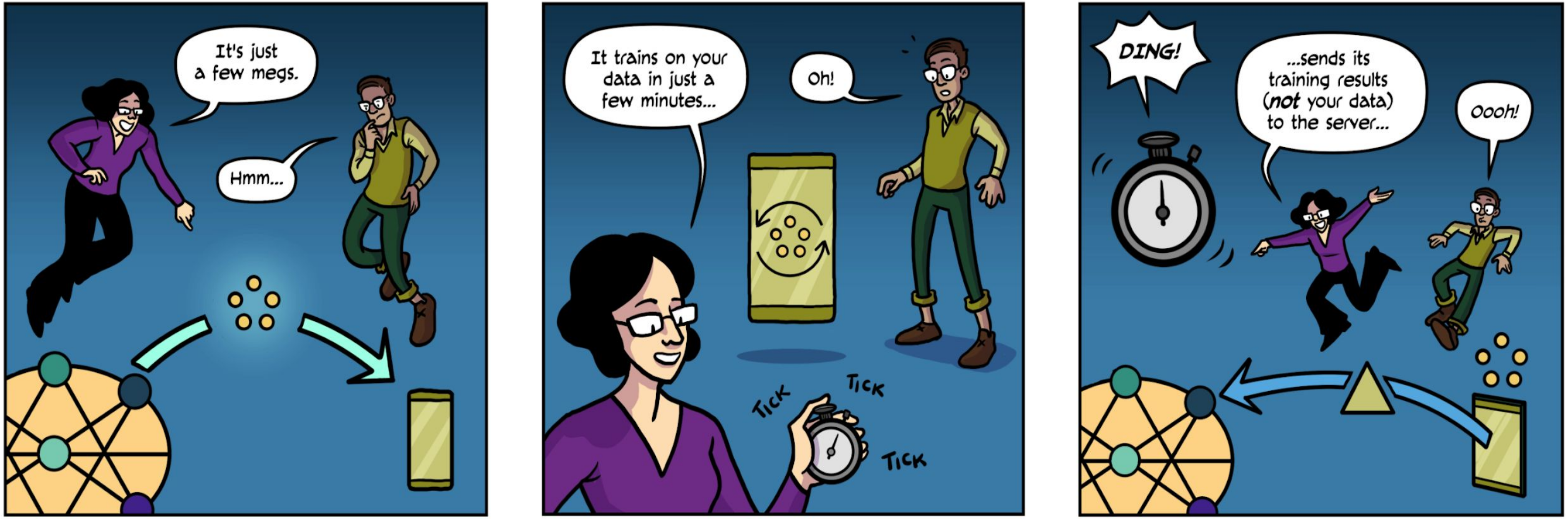
Federated Learning


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!*

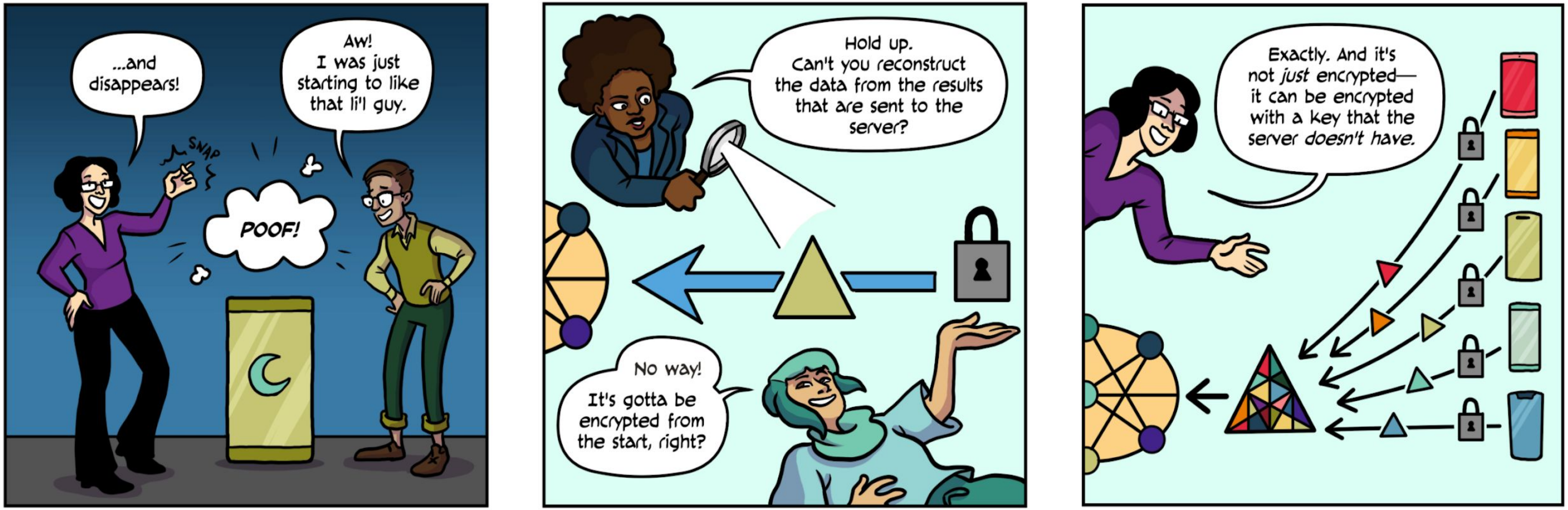
Federated Learning


Let's say you five represent all our users. Some of you probably have relevant data for the problem I'm trying to solve, and so—
Ooh! Pick me! Pick me! I'm eligible!
Uhh.
—and so, a subset of devices are selected—
Gyeeah!
—to receive a training model.

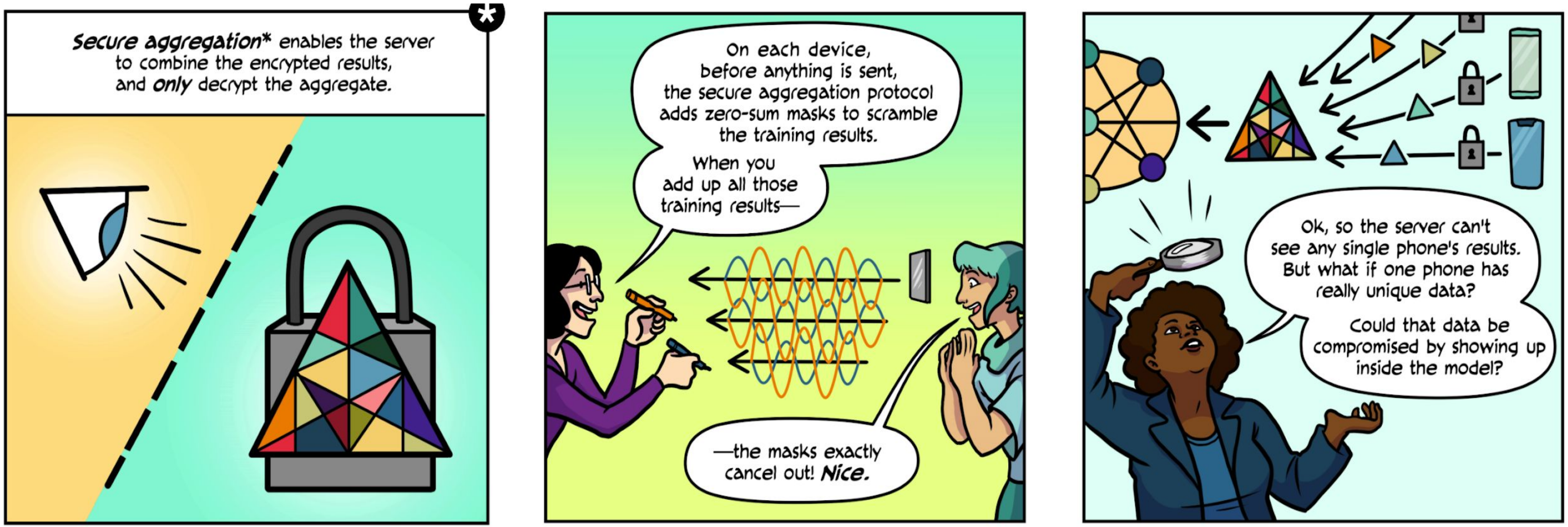
Federated Learning


It's just a few megs.
Hmm...
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

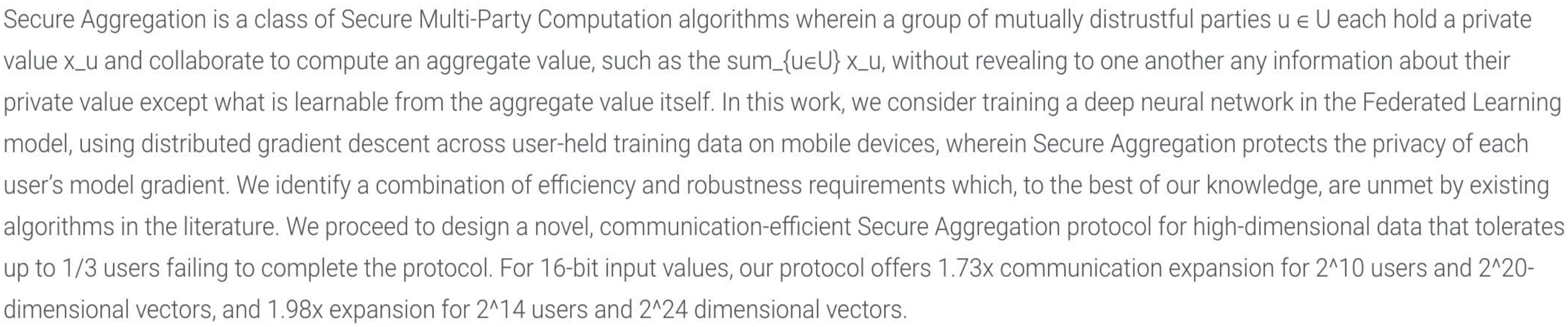

...and disappears!
SNAP
POOF!
Aw! I was just starting to like that li'l guy.
Hold up. Can't you reconstruct the data from the results that are sent to the server?
No way! It's gotta be encrypted from the start, right?
Exactly. And it's not *just* encrypted— it can be encrypted with a key that the server *doesn't have.*

Federated Learning


*
Secure aggregation* enables the server to combine the encrypted results, and only decrypt the aggregate.
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.
Ok, so the server can't see any single phone's results. But what if one phone has really unique data?
Could that data be compromised by showing up inside the model?

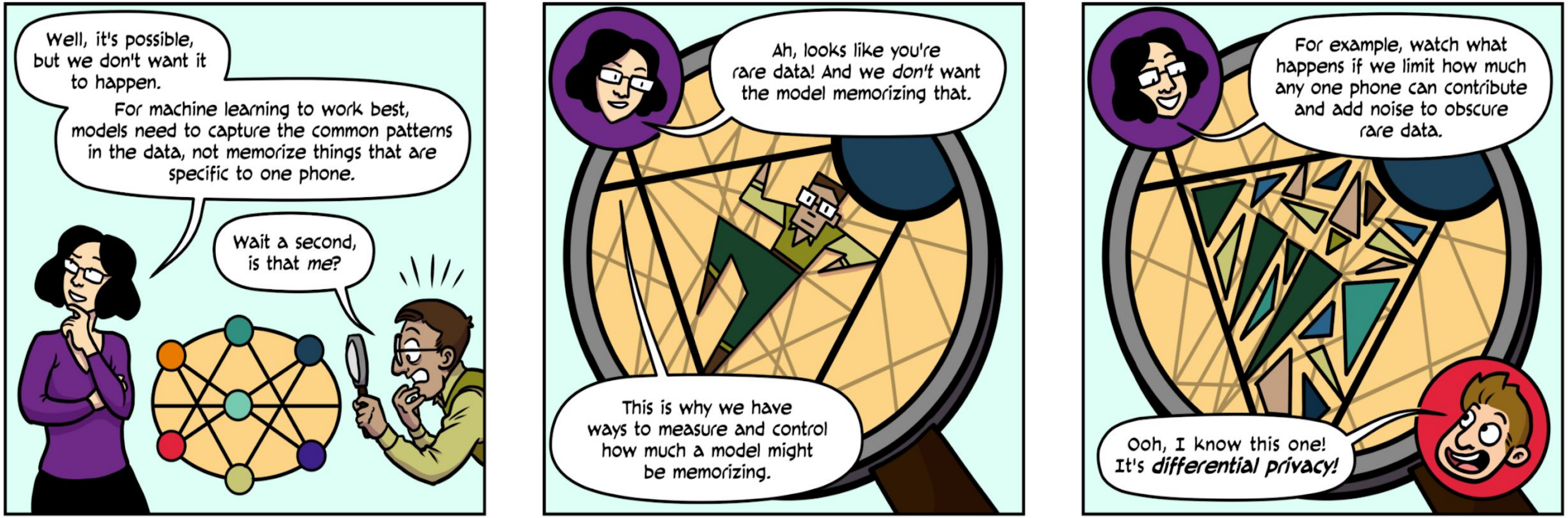
Surge aggregation

Source: https://research.google/pubs/pub45808/

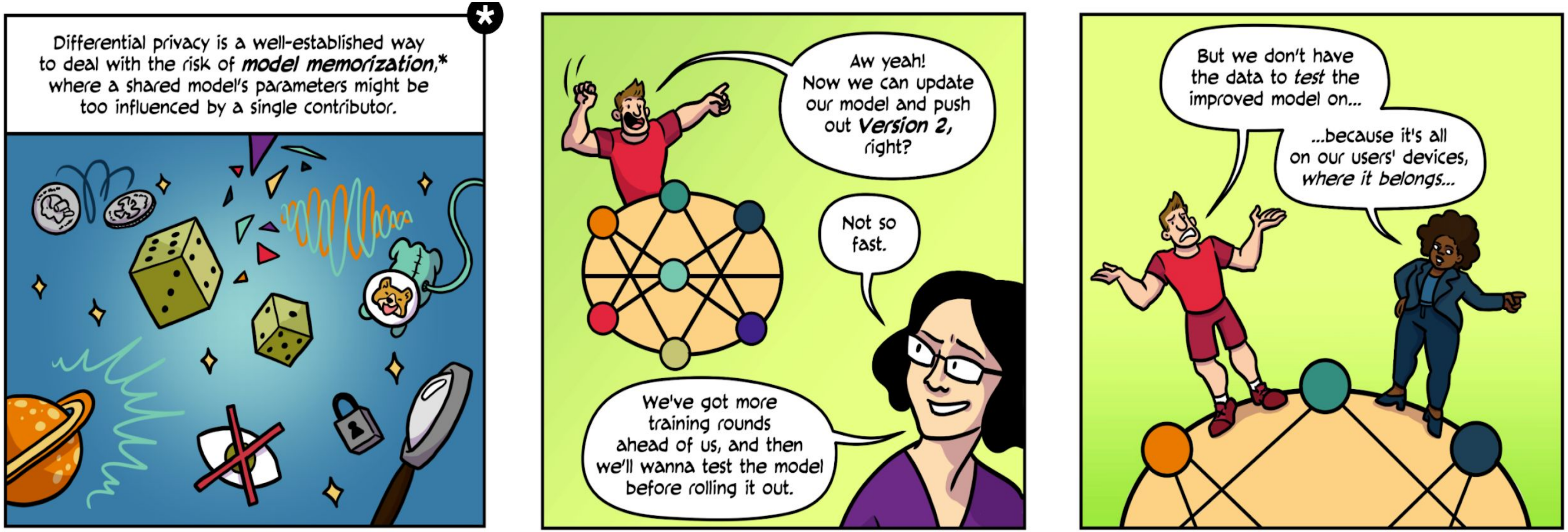


Secure Aggregation is a class of Secure Multi-Party Computation algorithms wherein a group of mutually distrustful parties u ∊ U each hold a private value x_u and collaborate to compute an aggregate value, such as the sum_{u∊U} x_u, without revealing to one another any information about their private value except what is learnable from the aggregate value itself. In this work, we consider training a deep neural network in the Federated Learning model, using distributed gradient descent across user-held training data on mobile devices, wherein Secure Aggregation protects the privacy of each user's model gradient. We identify a combination of efficiency and robustness requirements which, to the best of our knowledge, are unmet by existing algorithms in the literature. We proceed to design a novel, communication-efficient Secure Aggregation protocol for high-dimensional data that tolerates up to 1/3 users failing to complete the protocol. For 16-bit input values, our protocol offers 1.73x communication expansion for 2^10 users and 2^20-dimensional vectors, and 1.98x expansion for 2^14 users and 2^24 dimensional vectors.

Federated Learning


Well, it's possible, but we don't want it to happen.
For machine learning to work best, models need to capture the common patterns in the data, not memorize things that are specific to one phone.
Wait a second, is that *me*?
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!***

Federated Learning


Differential privacy is a well-established way to deal with the risk of *model memorization,** where a shared model's parameters might be too influenced by a single contributor.
*
Aw yeah! Now we can update our model and push out *Version 2,* right?
Not so fast.
We've got more training rounds ahead of us, and then we'll wanna test the model before rolling it out.
But we don't have the data to *test* the improved model on...
...because it's all on our users' devices, *where it belongs...*

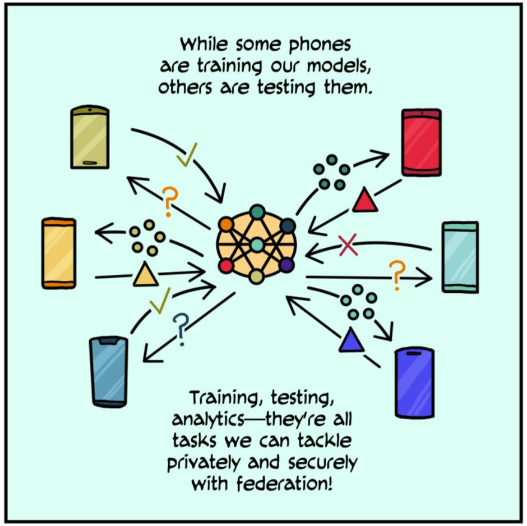
Memorization risk

Memorization risk can be mitigated by pre-filtering rare or sensitive information before training.

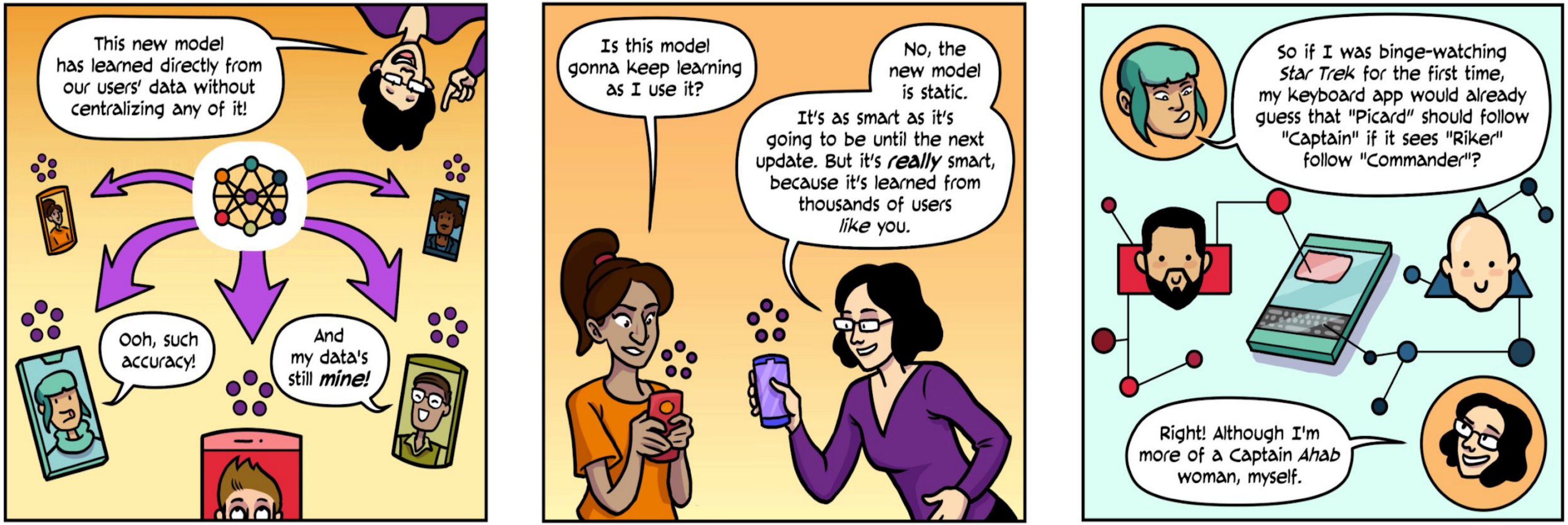
Source: https://federated.withgoogle.com/

Federated Learning

Yes! We safely trained ***on-device***, so now we can safely test the ***quality*** of that training where it matters most...
...in the hands of our users!


While some phones are training our models, others are testing them.
Training, testing, analytics—they're all tasks we can tackle privately and securely with federation!

Federated Learning


This new model has learned directly from our users' data without centralizing any of it!
Ooh, such accuracy!
And my data's still *mine!*
Is this model gonna keep learning as I use it?
No, the new model is static.
It's as smart as it's going to be until the next update. But it's ***really*** smart, because it's learned from thousands of users *like* you.
So if I was binge-watching *Star Trek* for the first time, my keyboard app would already guess that "Picard" should follow "Captain" if it sees "Riker" follow "Commander"?
Right! Although I'm more of a Captain *Ahab* woman, myself.

Machine Learning & Decentralization

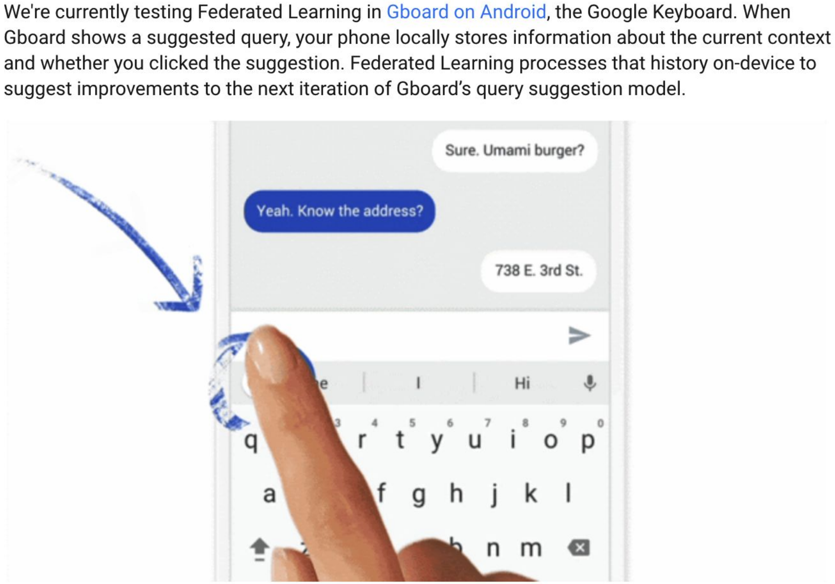
- We know the advantages of **AI** (see #CircleSummi1)
- We know the advantages of **decentralization** (see #CircleNight4)
- But of course, apply AI to every project is not useful…
- Apply the decentralization to every project is not useful…
- What do I mean?

Machine Learning & Decentralization

- We have to integrate the appropriate technology when:
  - It is strictly necessary
  - When we want to innovate a product/service
  - When we want to innovate an economic sector
  - When we want to guarantee the users’ privacy

Machine Learning & Decentralization

- An example: Gboard on Android



We're currently testing Federated Learning in Gboard on Android, the Google Keyboard. When Gboard shows a suggested query, your phone locally stores information about the current context and whether you clicked the suggestion. Federated Learning processes that history on-device to suggest improvements to the next iteration of Gboard's query suggestion model.

Sure. Umami burger?

Yeah. Know the address?

738 E. 3rd St.

Thanks!

AI & Decentralization lover