Google

Federated Learning

Privacy-Preserving Collaborative Machine Learning without Centralized Training Data

Jakub Konečný
konkey@google.com

presenting the work of many

Trends in Optimization Seminar, University of Washington in Seattle
Jan 30, 2018

This Talk

1. **Why Federated Learning?**
2. FL vs. other distributed optimization
3. FL Algorithms
4. Minimizing Communication
5. Secure Aggregation
6. Differential Privacy

Federated Learning

Our Goal

Imbue **mobile devices** with **state of the art machine learning** systems **without centralizing data** and **with privacy** by default.

Federated Learning

Our Goal

Imbue **mobile devices** with **state of the art machine learning** systems **without centralizing data** and **with privacy** by default.

A very *personal* computer

2015: 79% away from phone ≤2 hours/day[1]
63% away from phone ≤1 hour/day
25% can't remember being away at all

2013: 72% of users within 5 feet of phone most of the time[2].

Plethora of sensors

Innumerable digital interactions

[1] 2015 Always Connected Research Report, IDC and Facebook
[2] 2013 Mobile Consumer Habits Study, Jumio and Harris Interactive.

Federated Learning

Our Goal

Imbue **mobile devices** with **state of the art machine learning** systems **without centralizing data** and **with privacy** by default.

Deep Learning

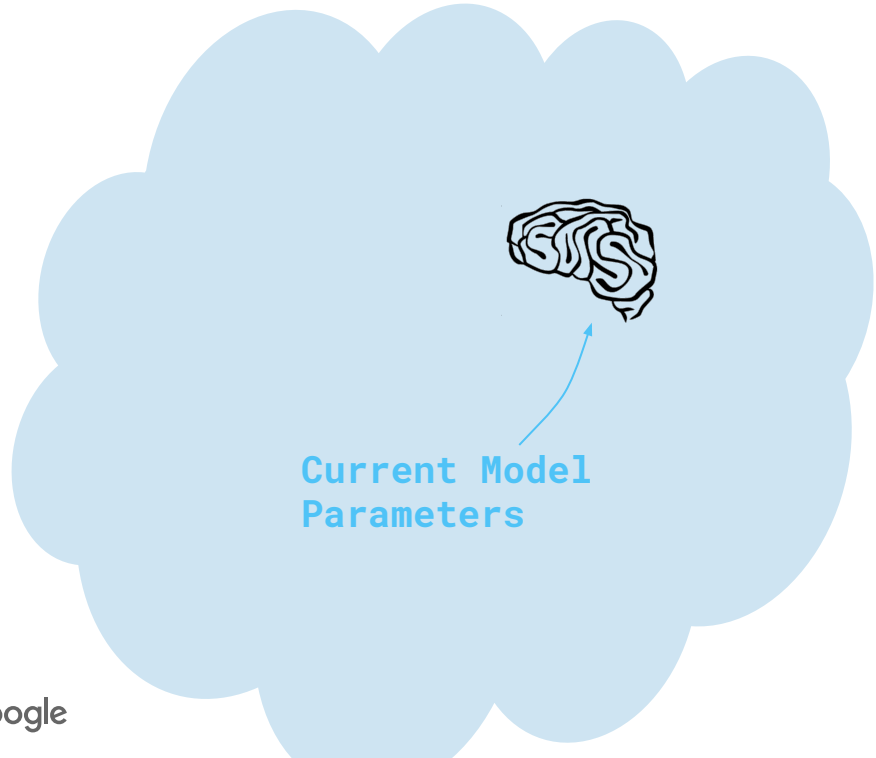
non-convex

millions of parameters

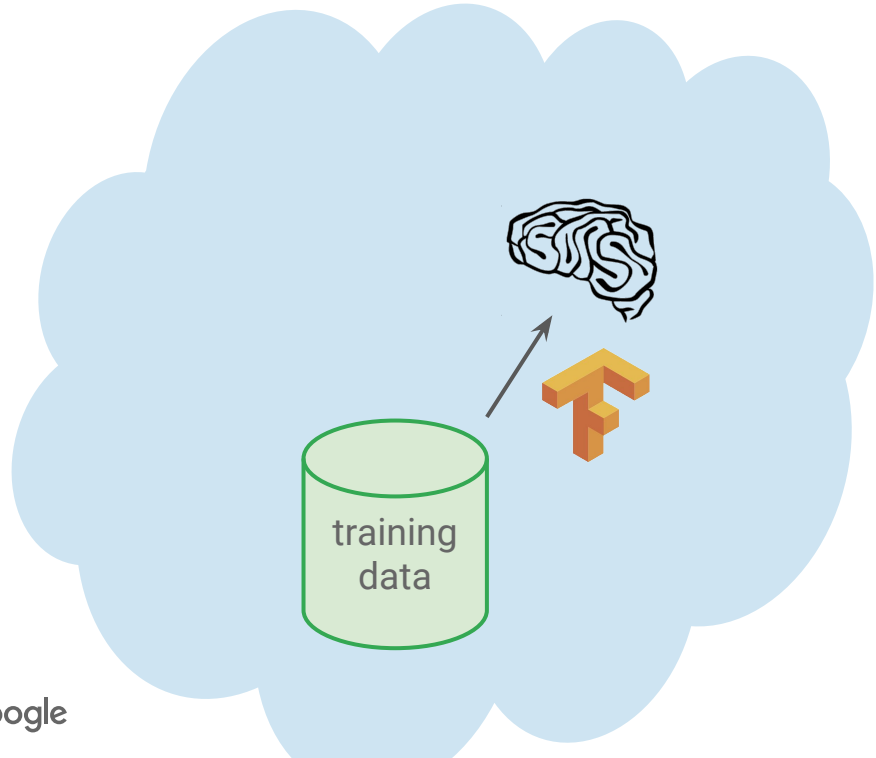
complex structure (eg LSTMs)

Cloud-centric ML for Mobile

The model lives in the cloud.



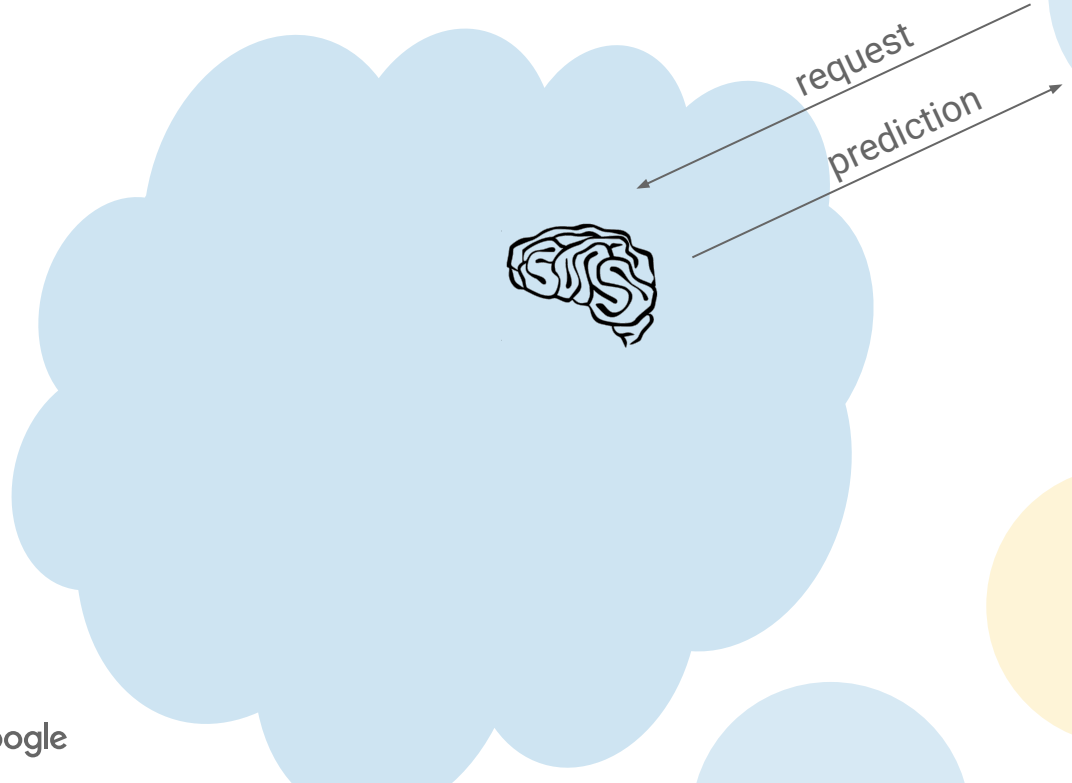
We train models in the cloud.



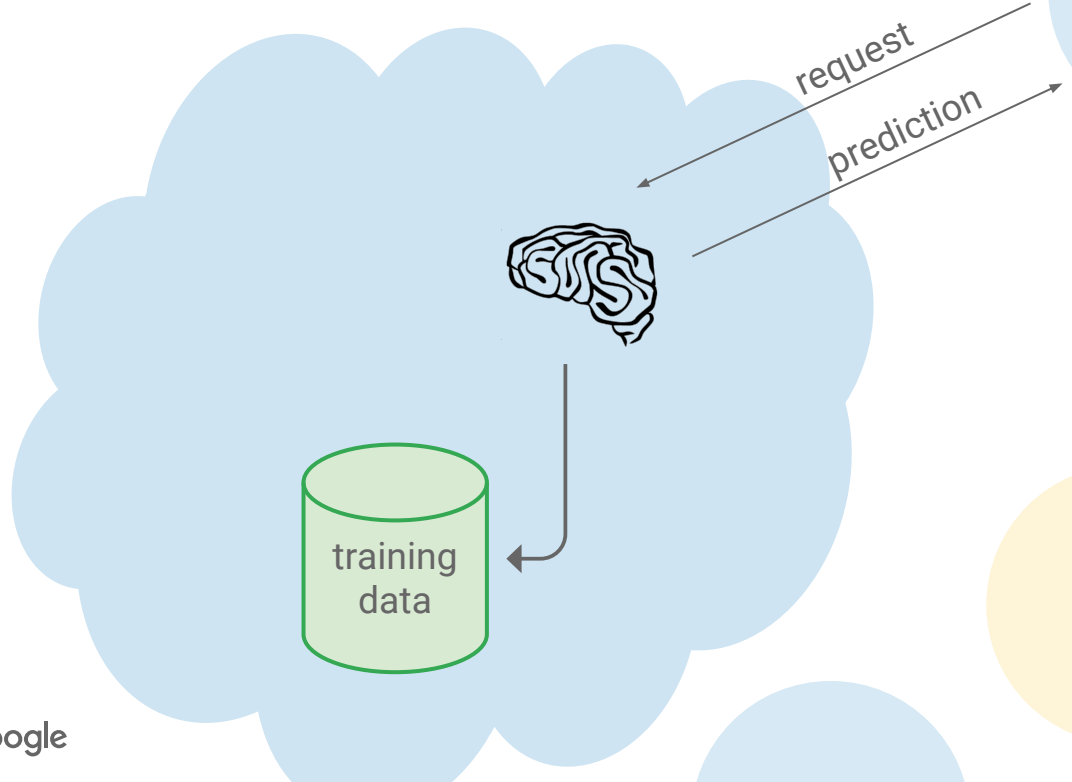
Mobile Device

Current Model Parameters

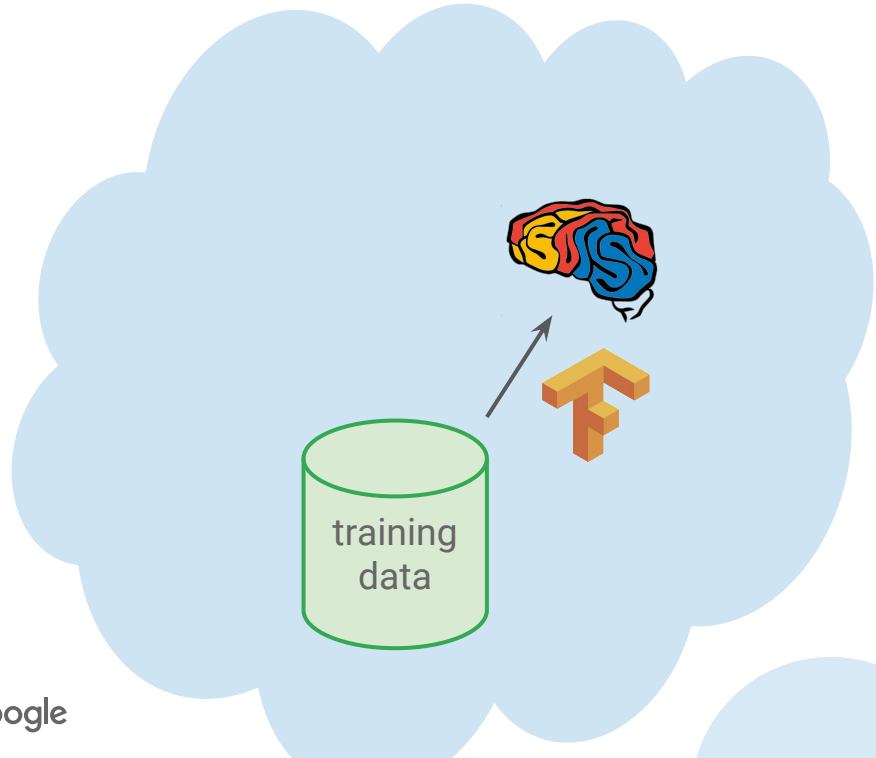
Make predictions in the cloud.



Gather training data in the cloud.



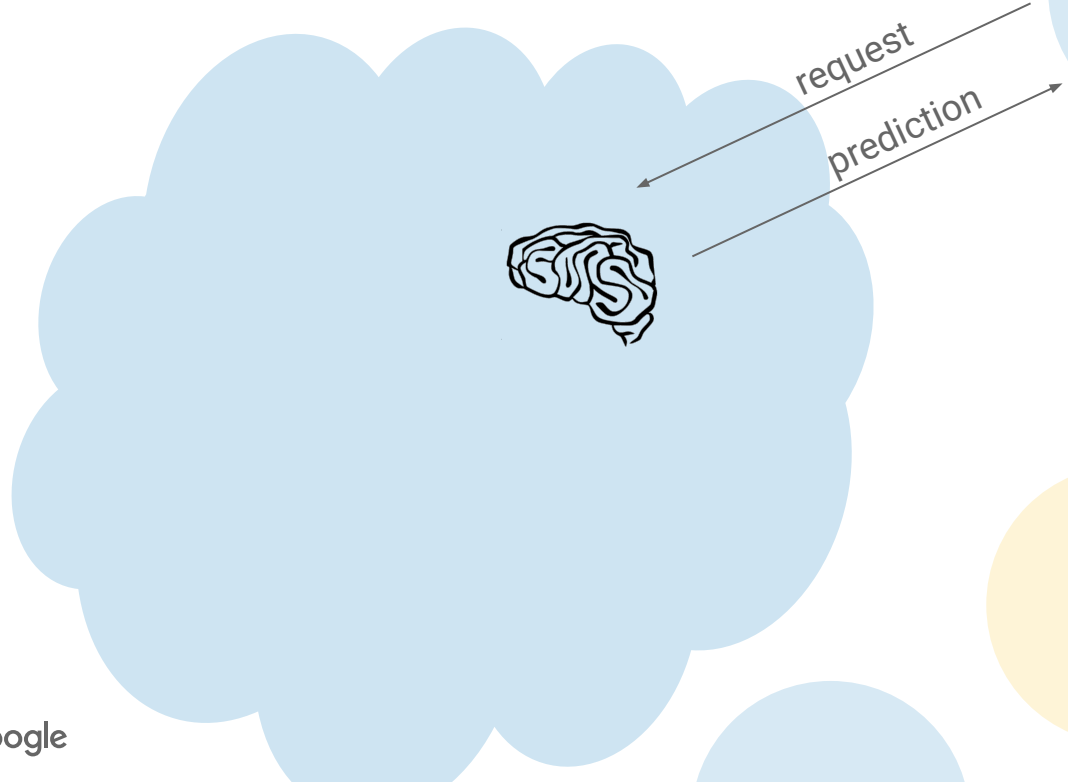
And make the models better.



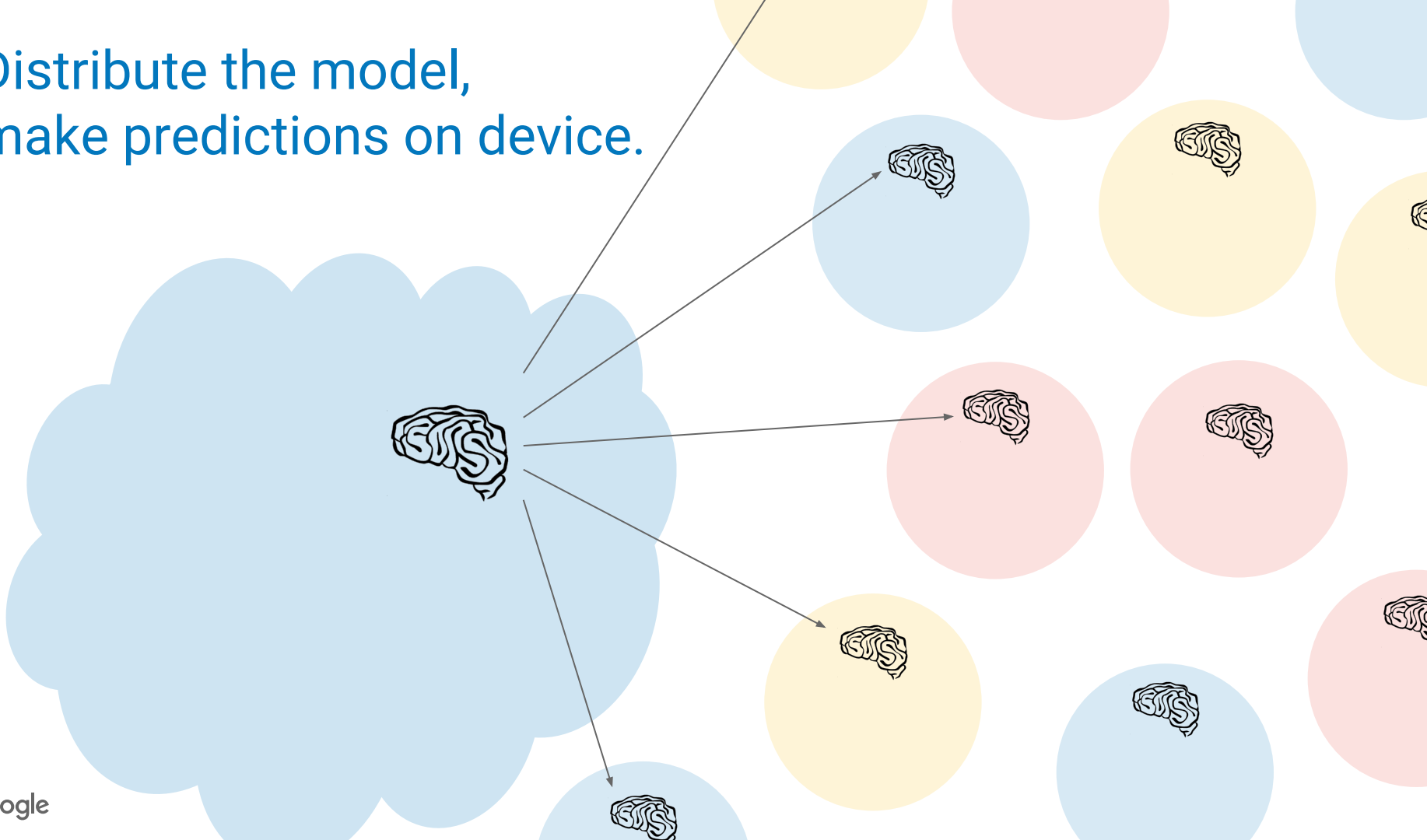
On-device inference

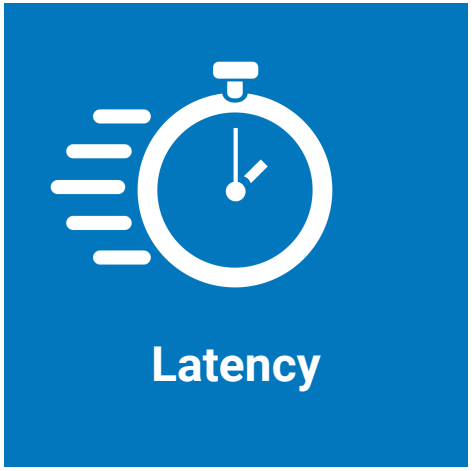
On-device inference is using a cloud-distributed model to make predictions directly on an edge device without a cloud round-trip.

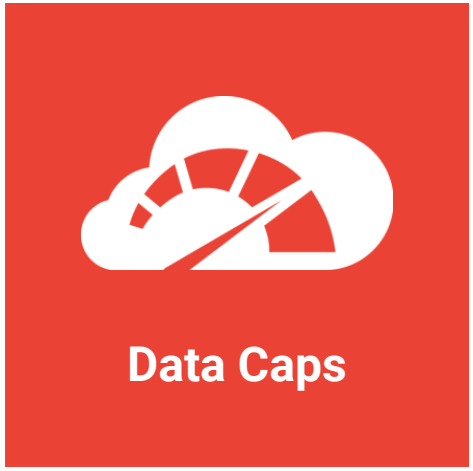
Instead of making predictions in the cloud



Distribute the model, make predictions on device.



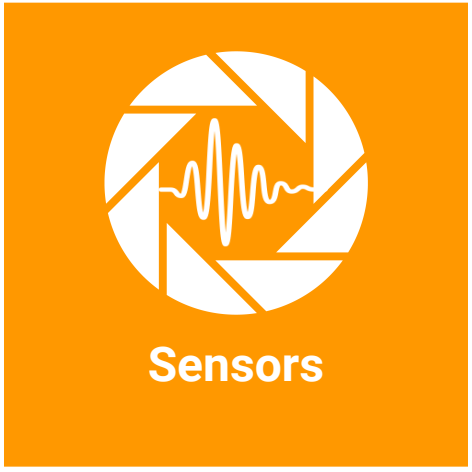

Latency


Data Caps

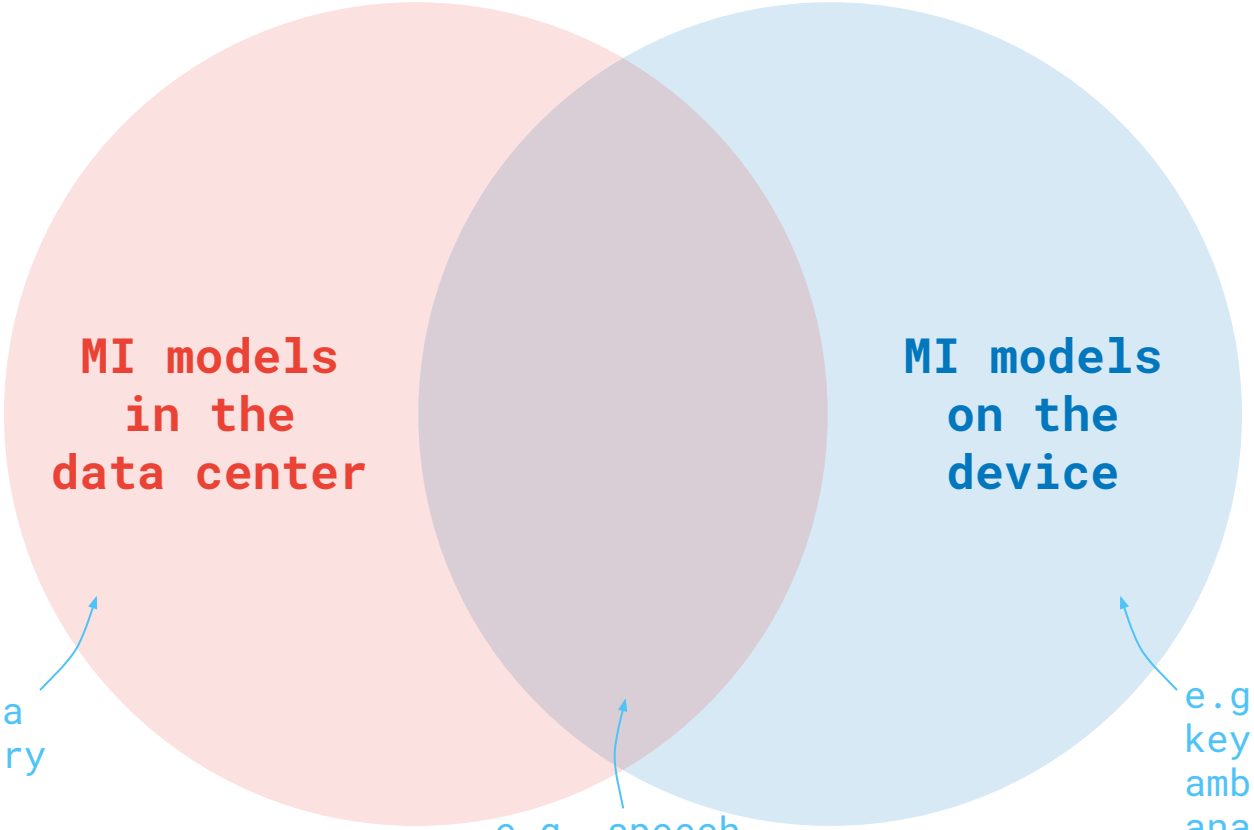

Privacy


Offline

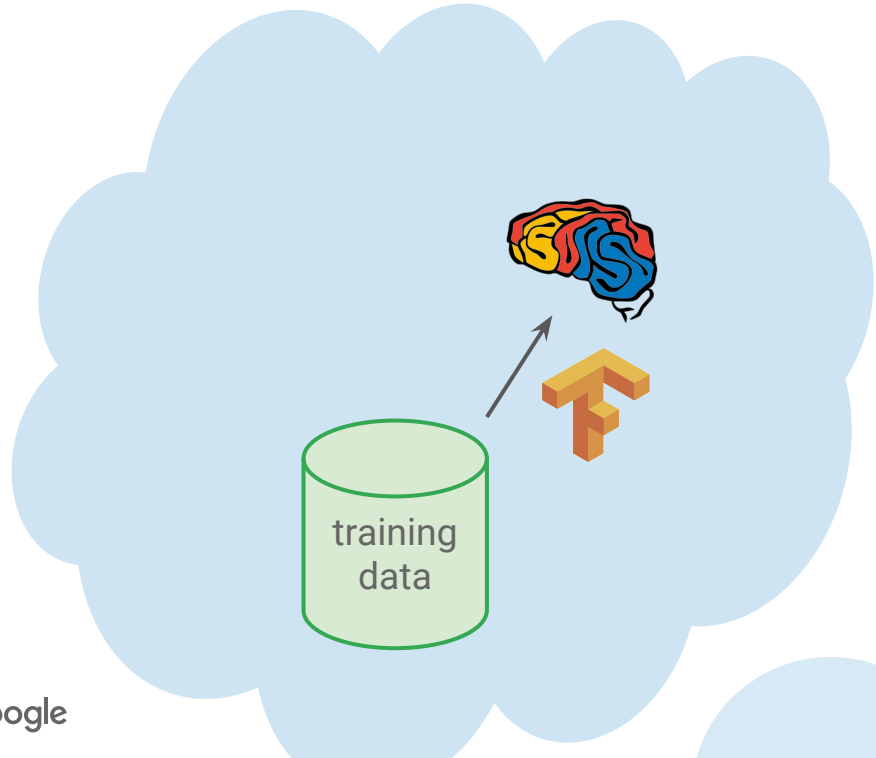

Power


Sensors

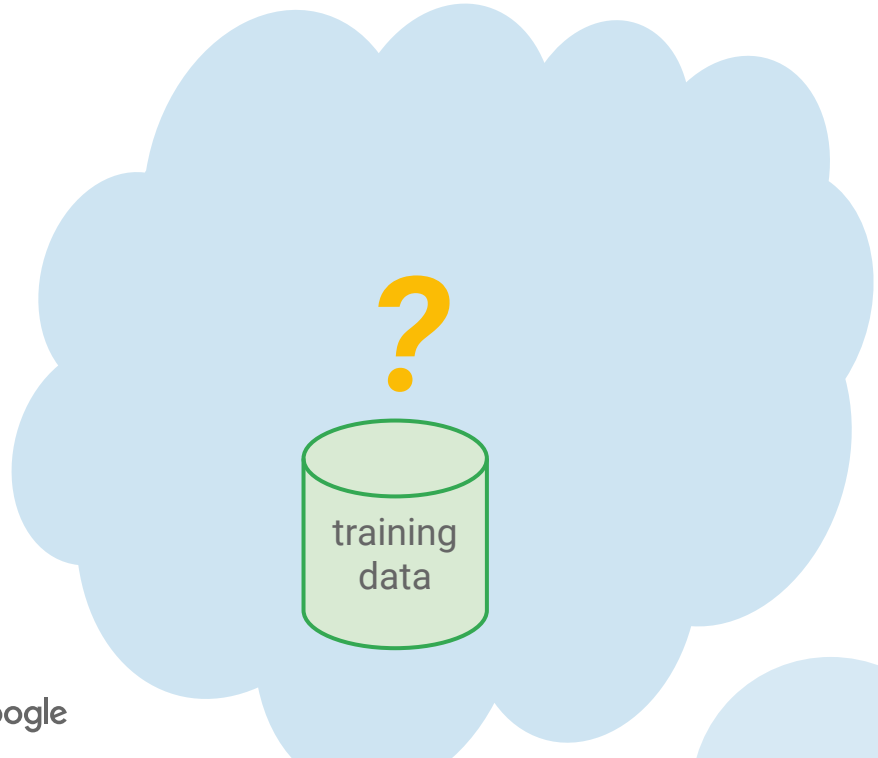
Machine Intelligence for Mobile Devices



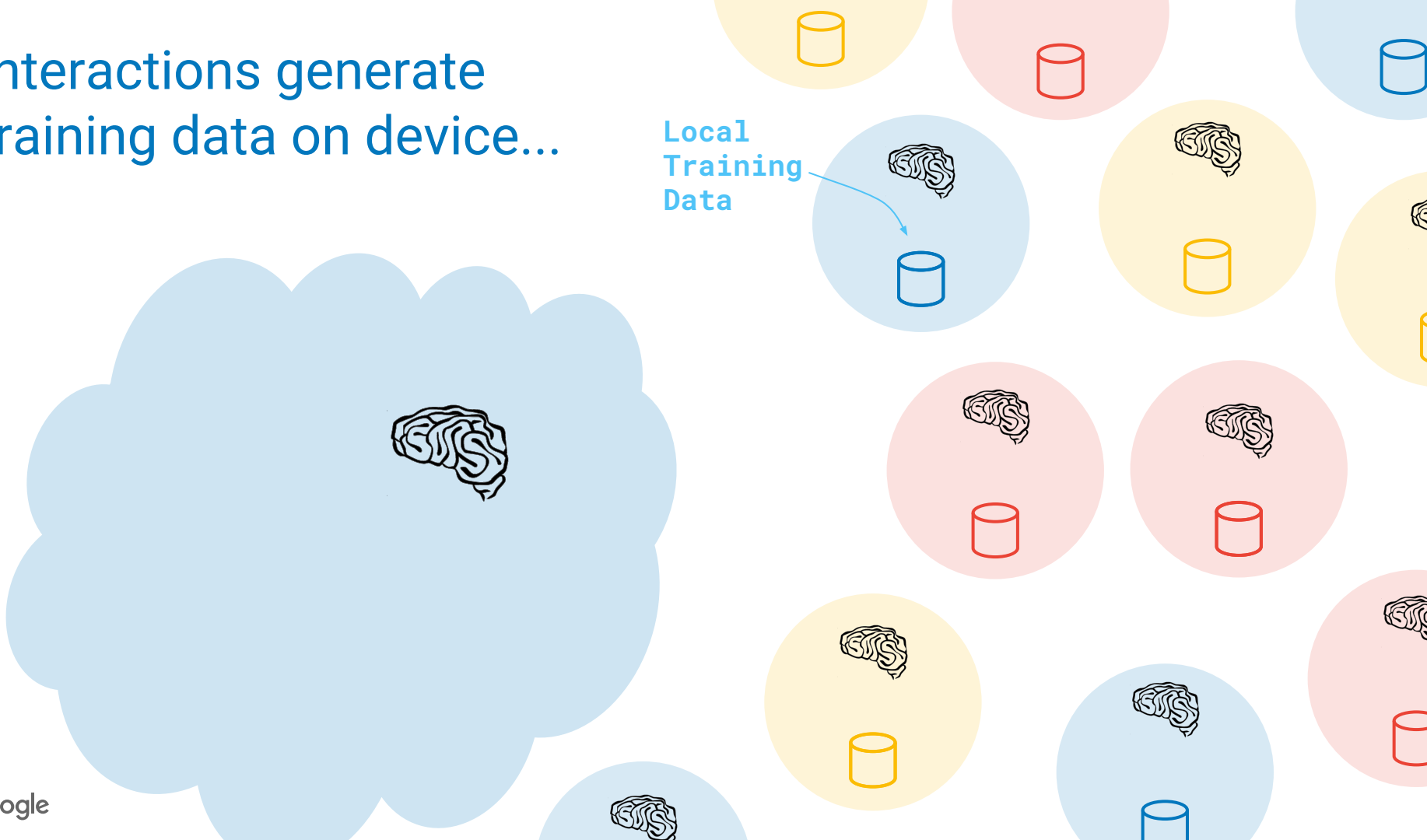
But how do we continue to improve the model?



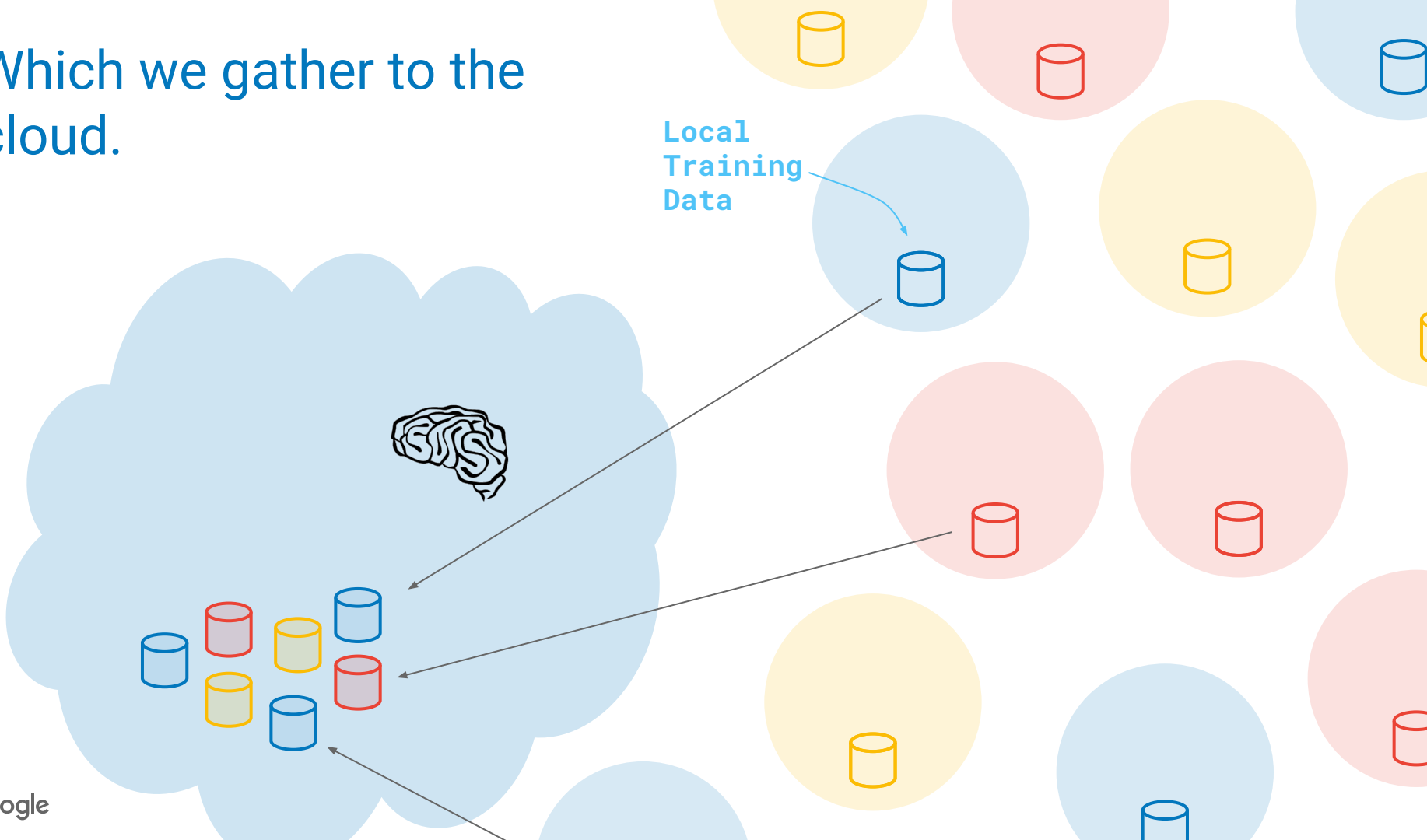
But how do we continue to improve the model?



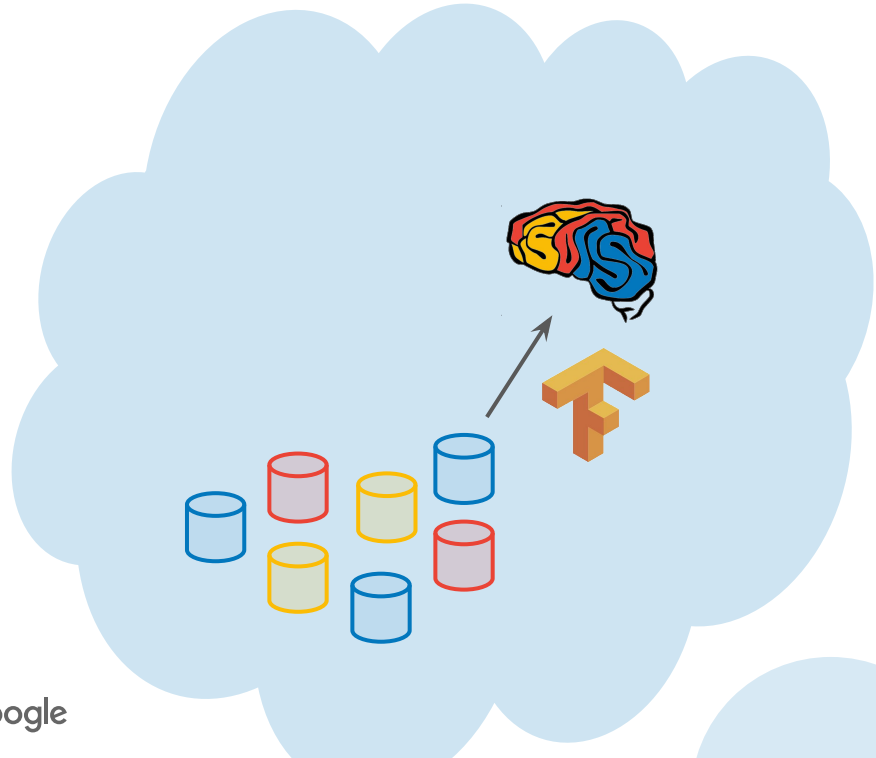
Interactions generate training data on device...



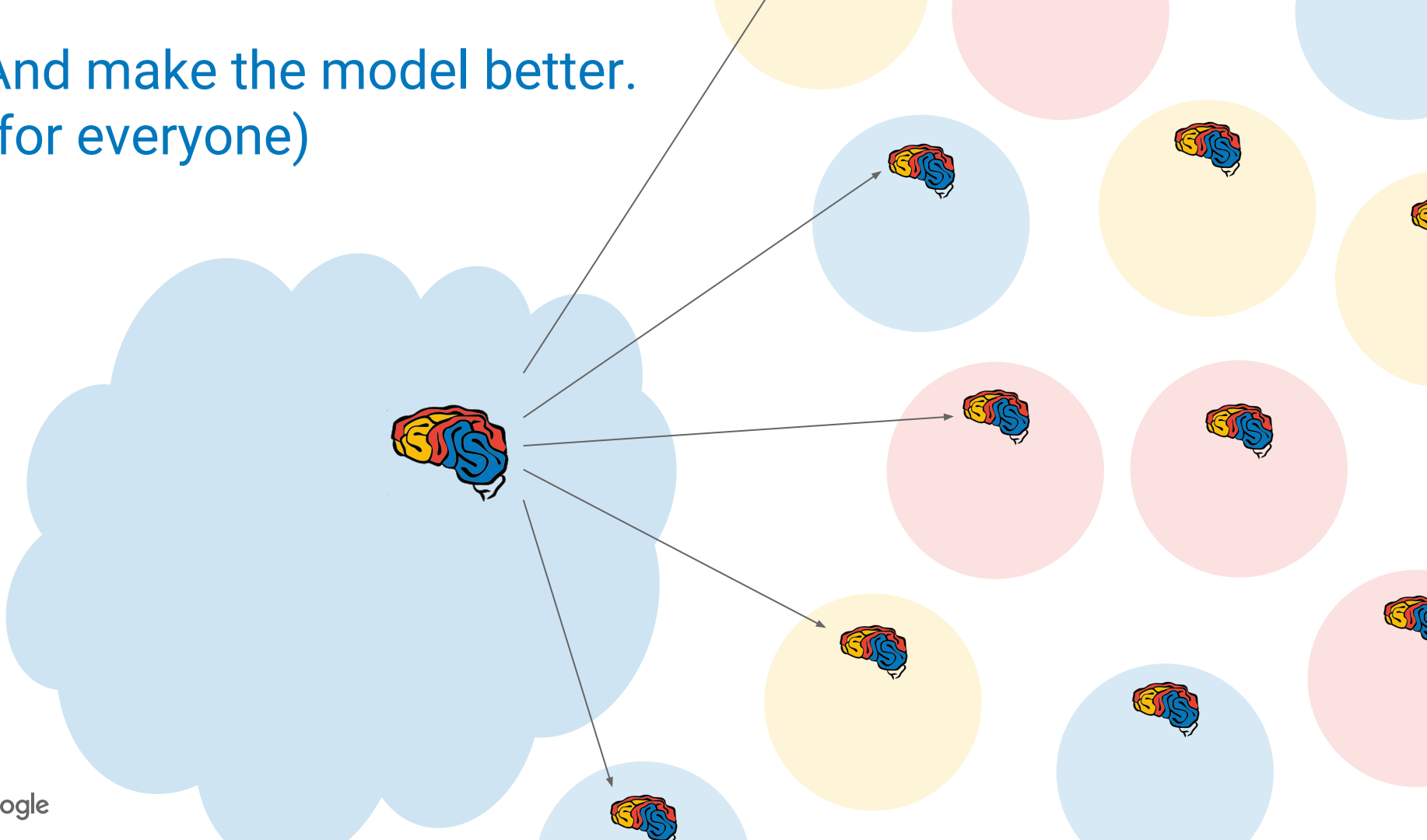
Which we gather to the cloud.

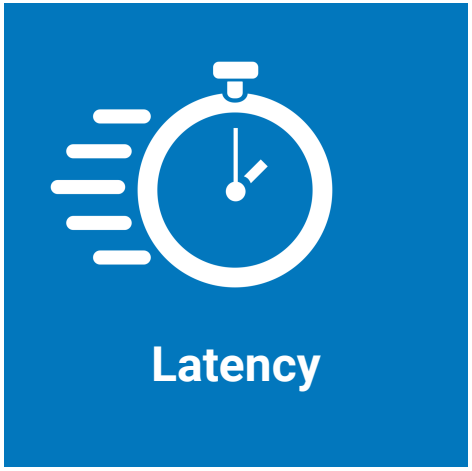


And make the model better.



And make the model better. (for everyone)




Latency

Data Caps

Privacy

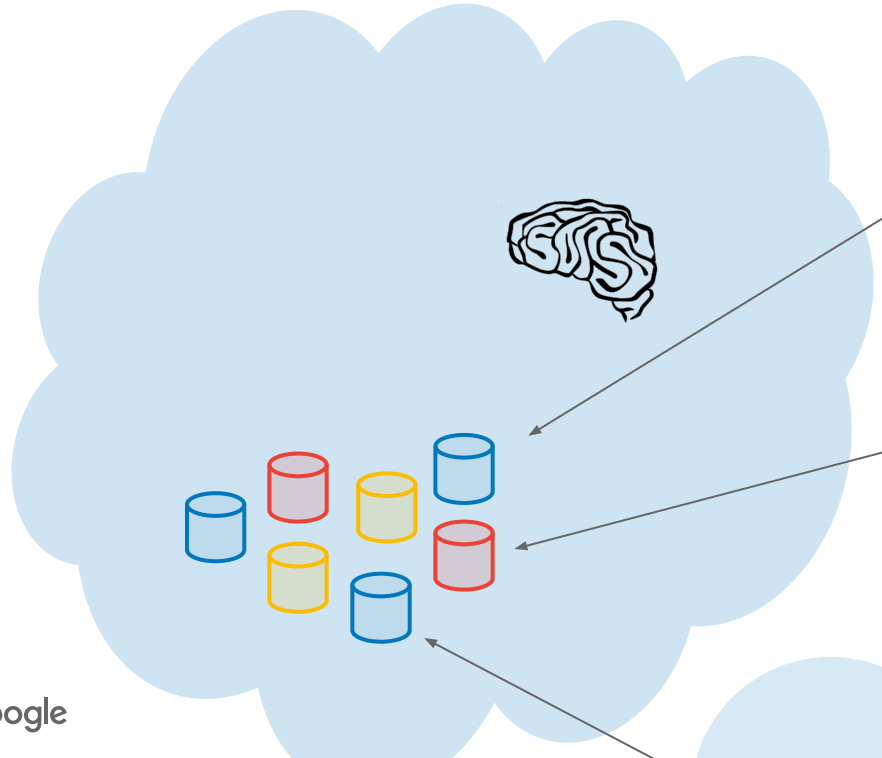

Offline

Power

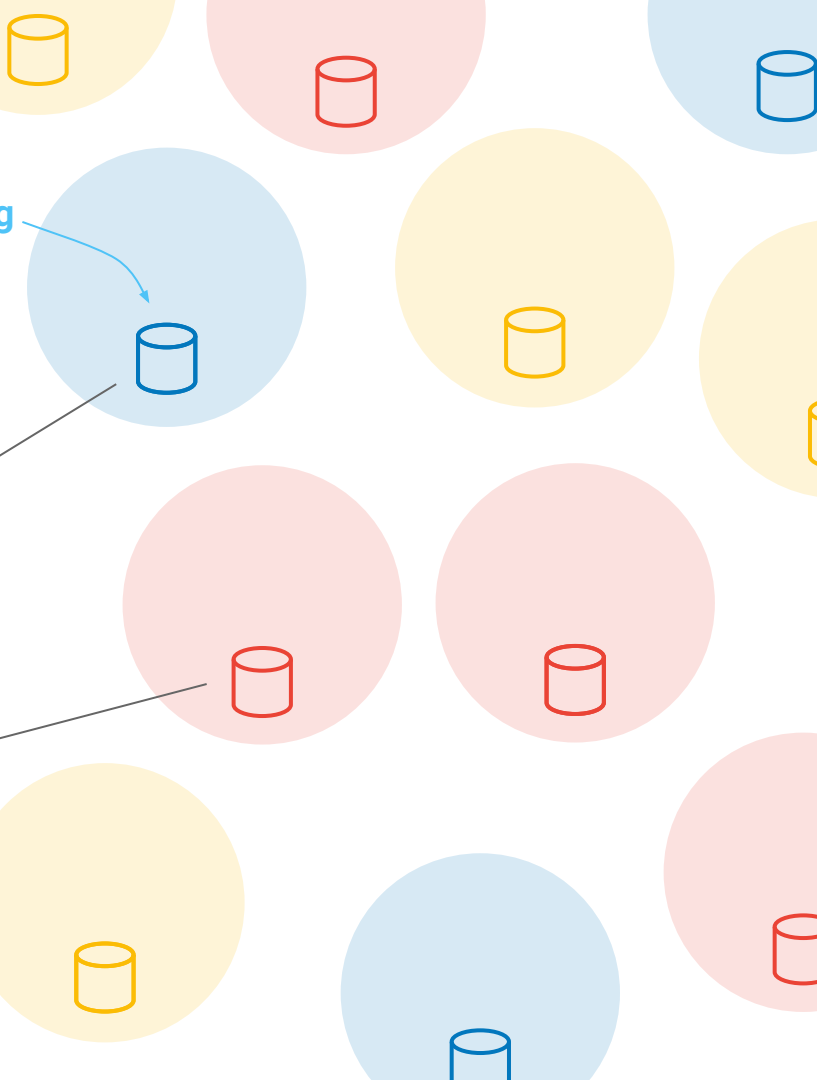
Sensors

On-Device **Learning**
(Personalization)

Instead of centralizing the training data...

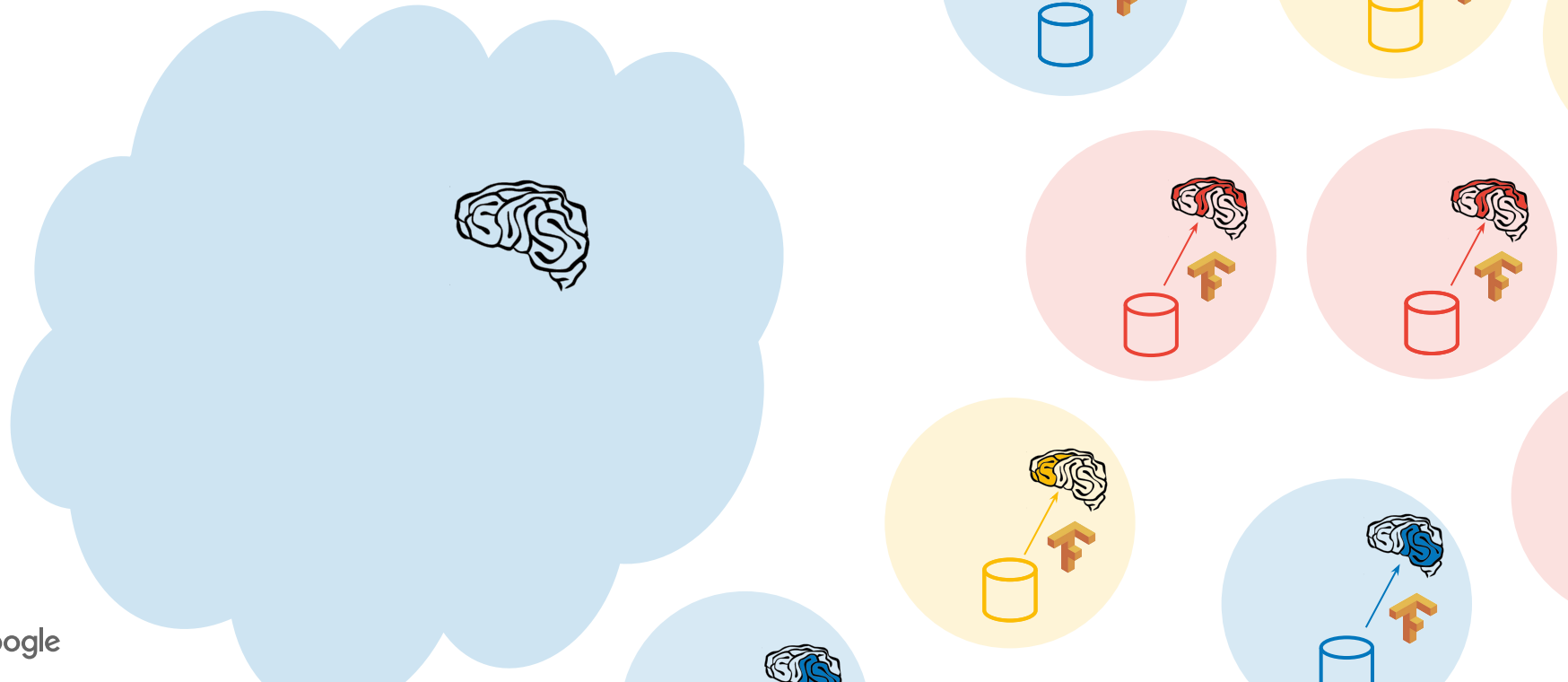


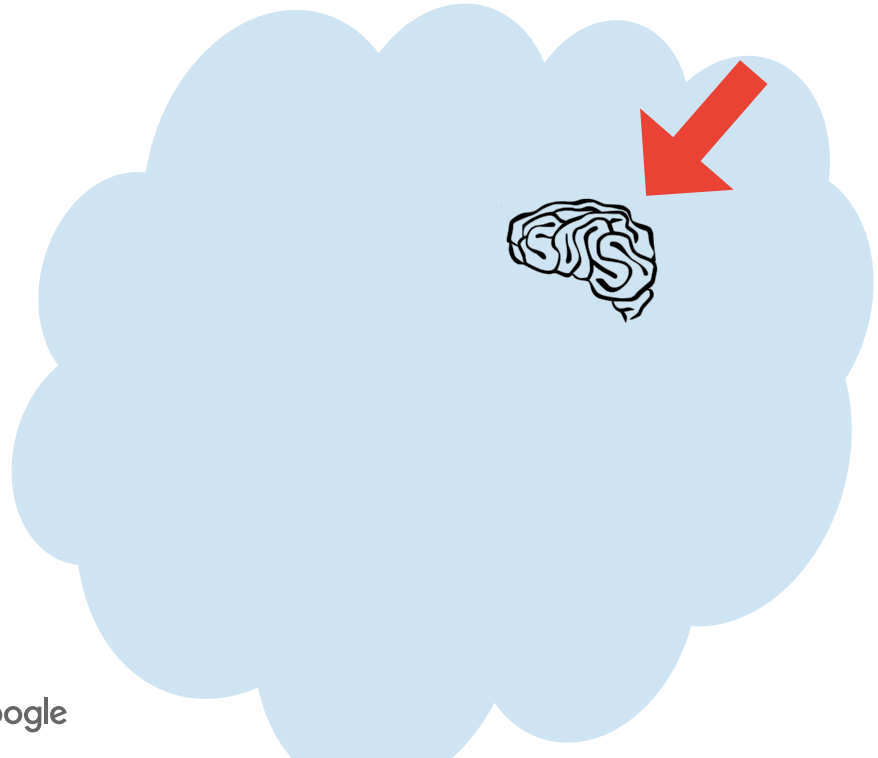
Local Training Data



Google

Train models right on the device.
Better for everyone (individually.)





But what about…

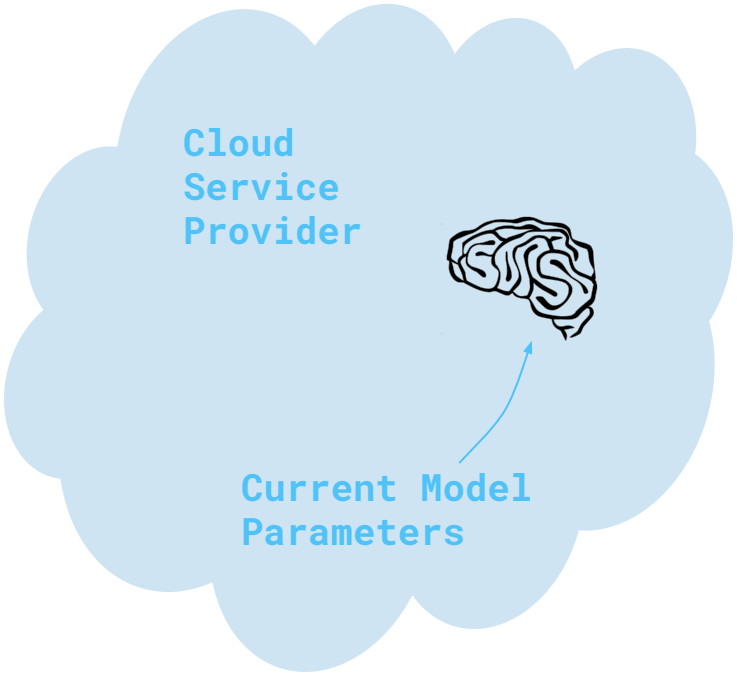
1. New User Experience
2. Benefitting from peers' data

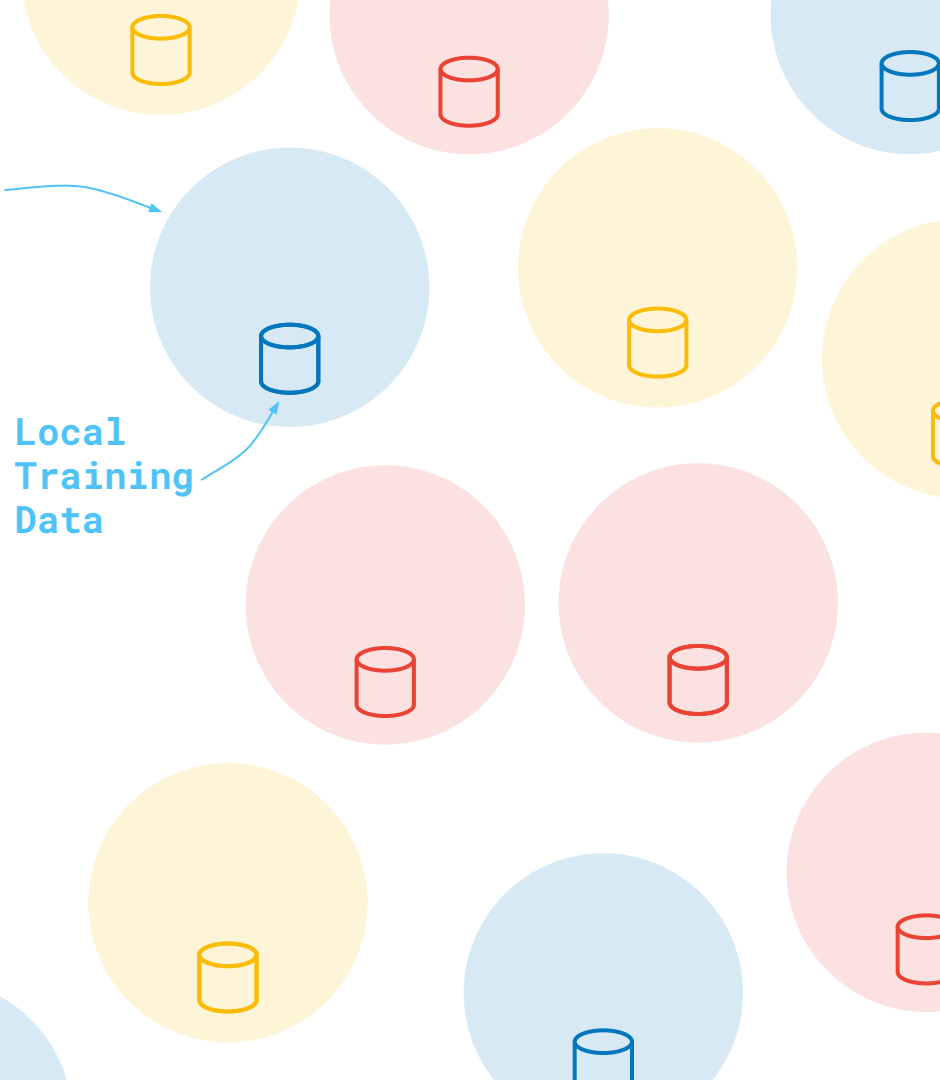
Federated computation and learning

Federated computation: where a server coordinates a fleet of participating devices to compute aggregations of devices' private data.

Federated learning: where a shared global model is trained via federated computation.

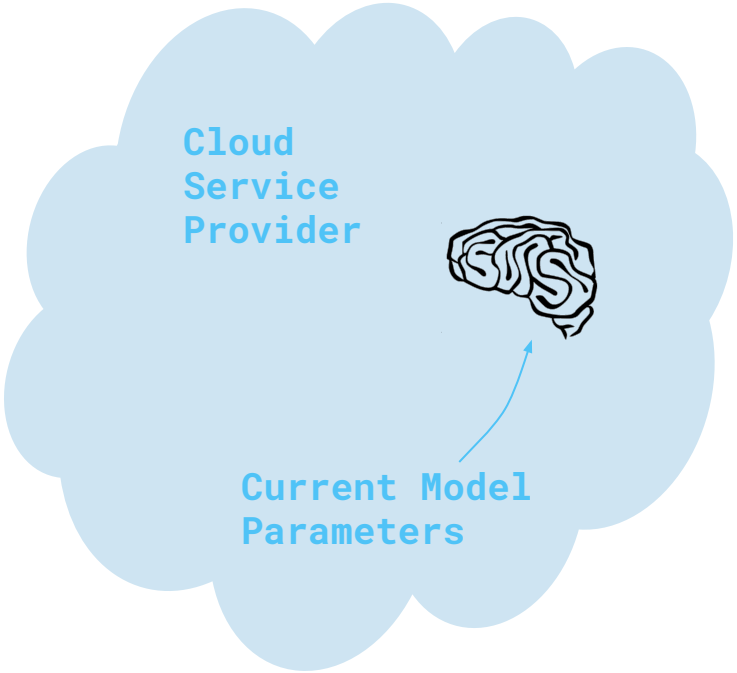
Federated Learning

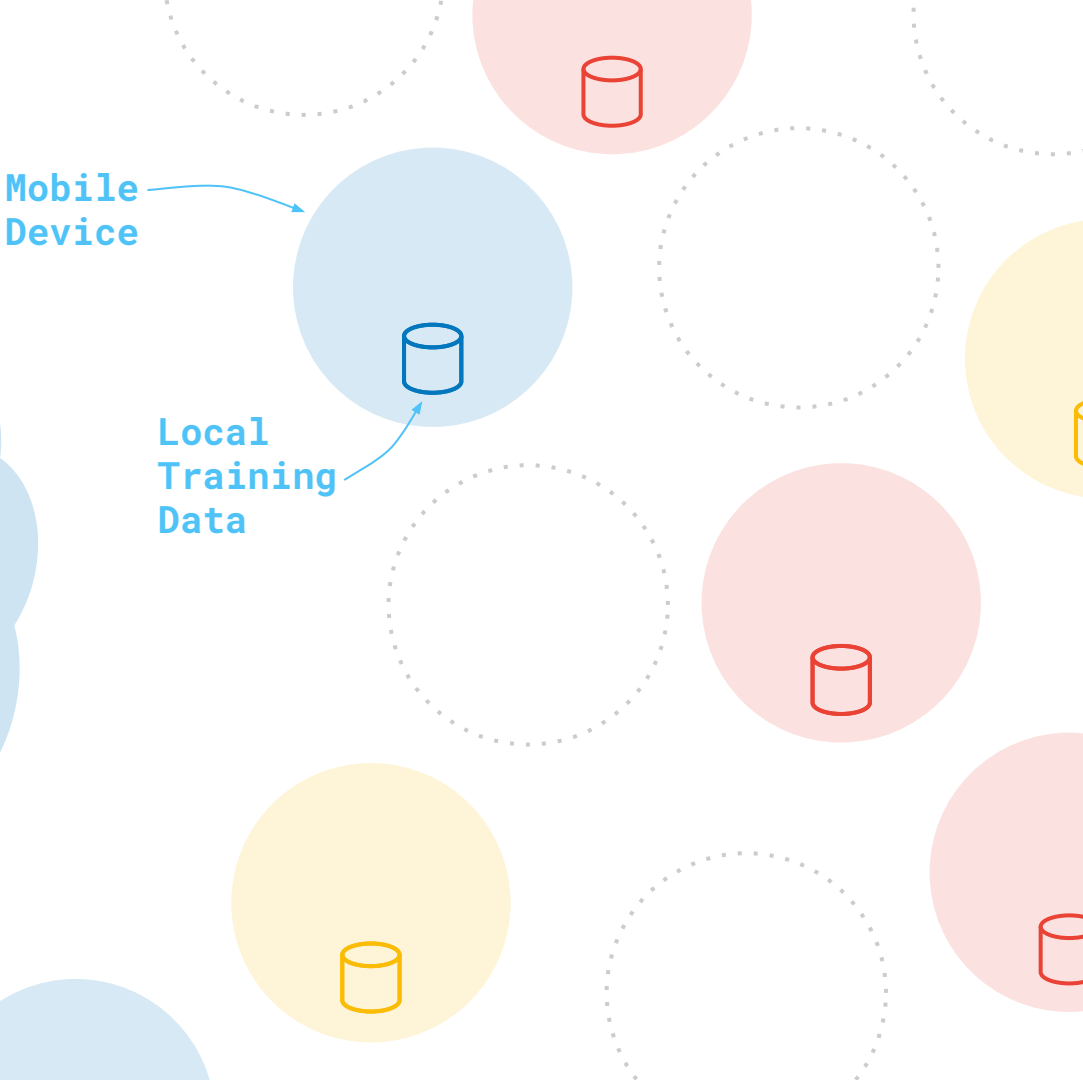




Federated Learning

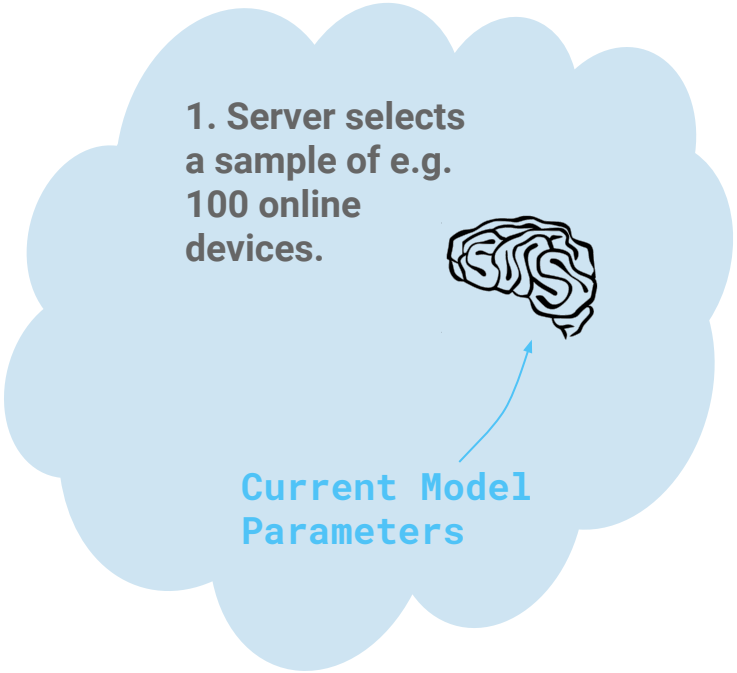
Many devices will be offline.

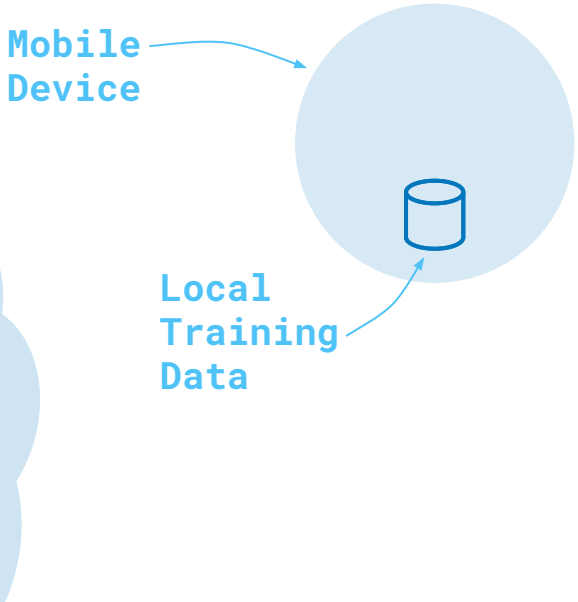


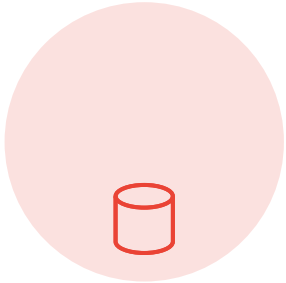


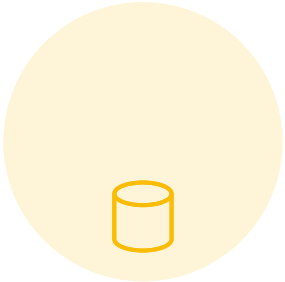
Federated Learning

Many devices will be offline.

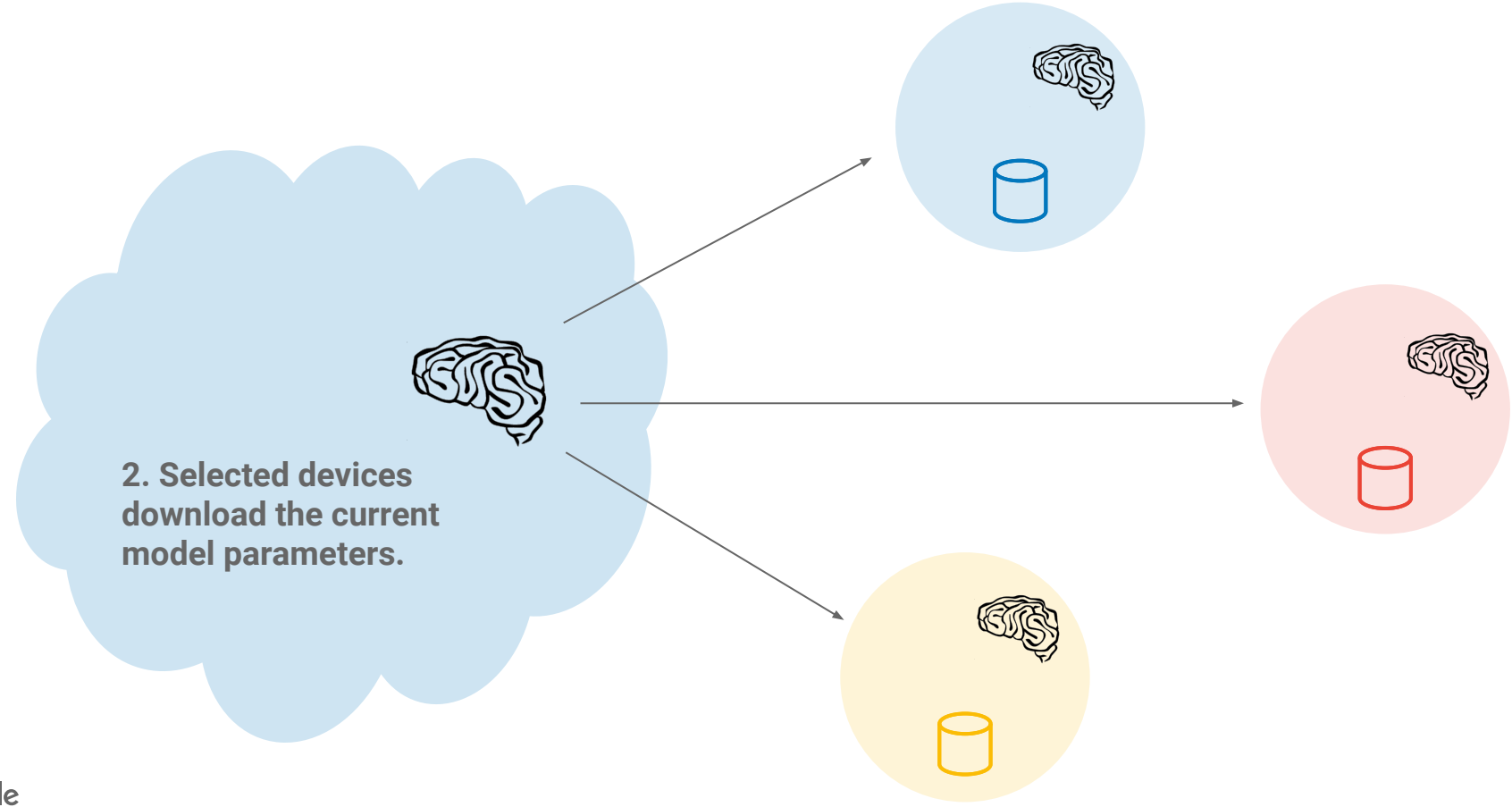




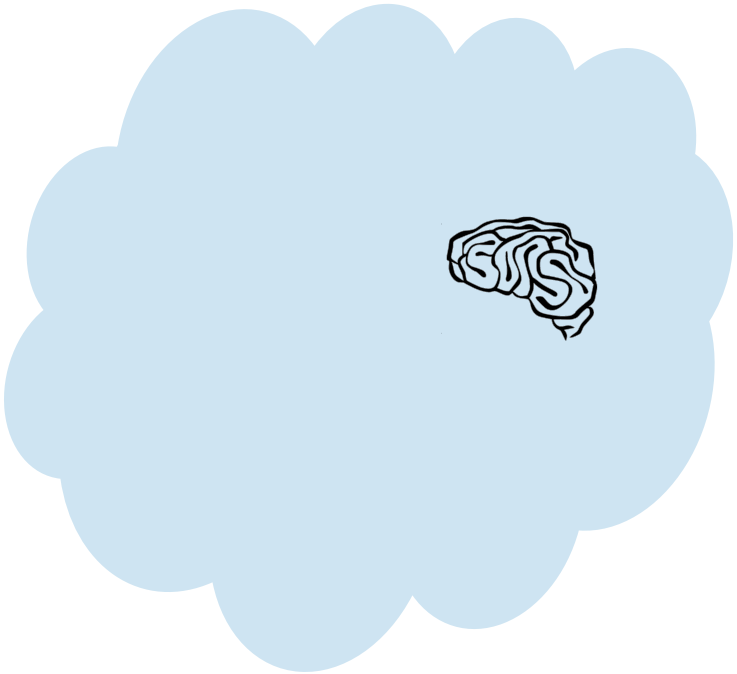


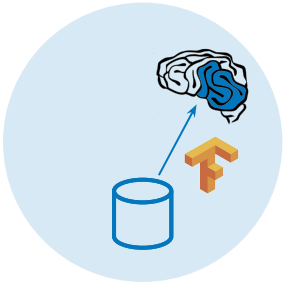


Federated Learning

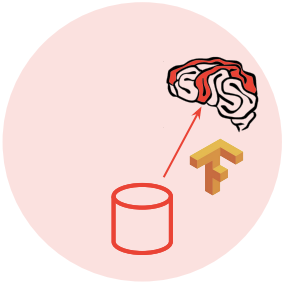


Federated Learning

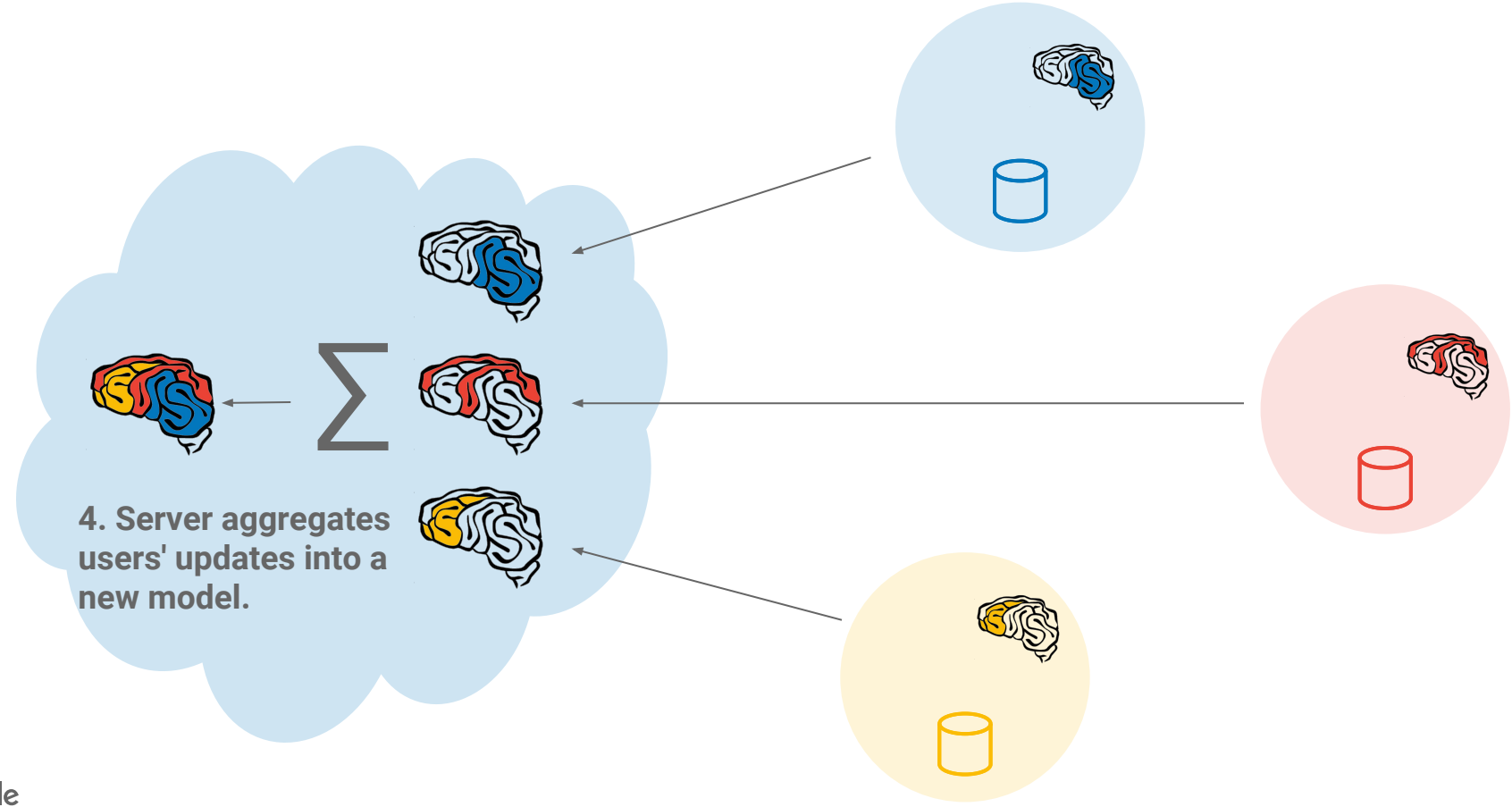




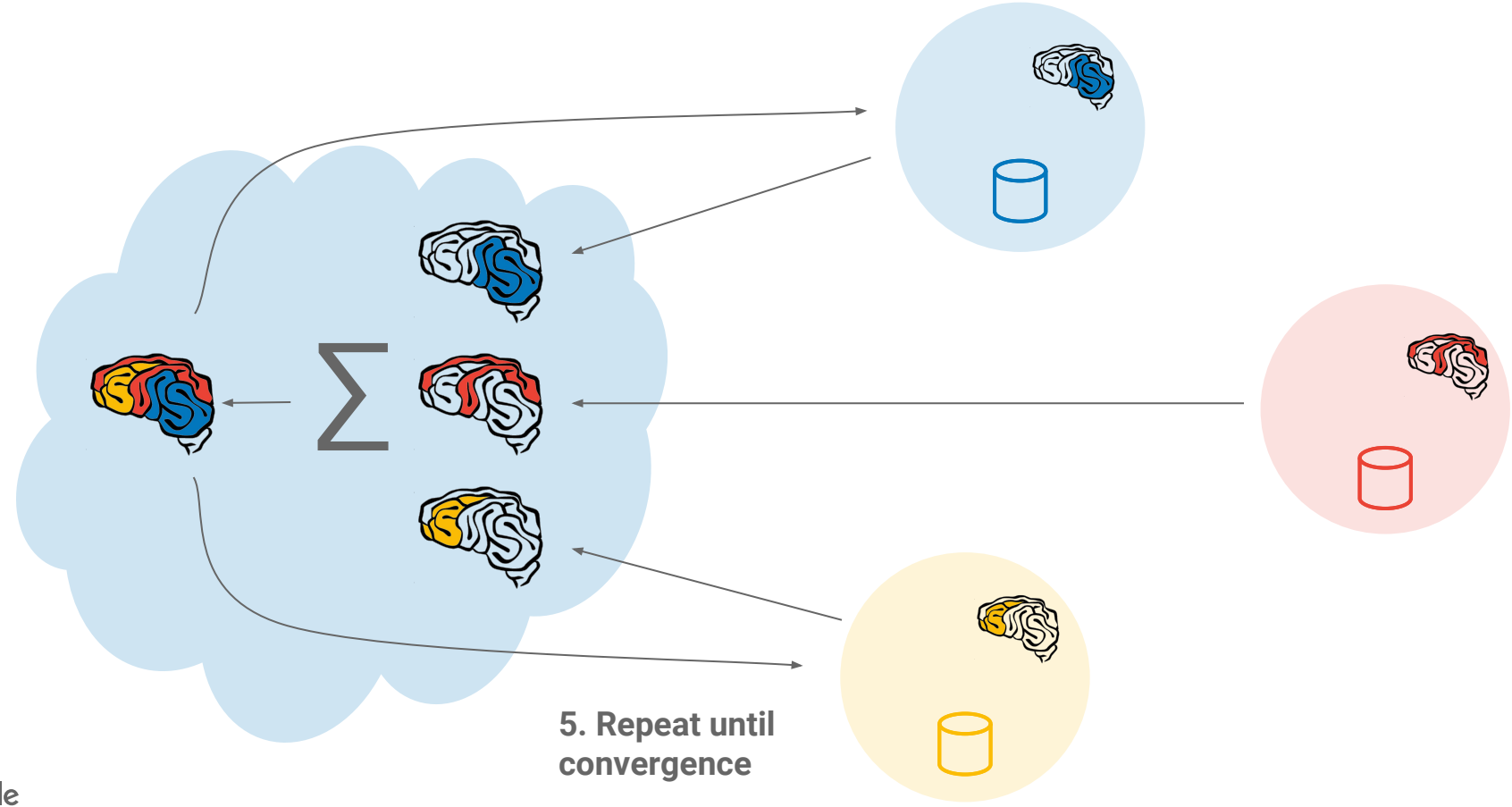
3. Devices compute an update using local training data



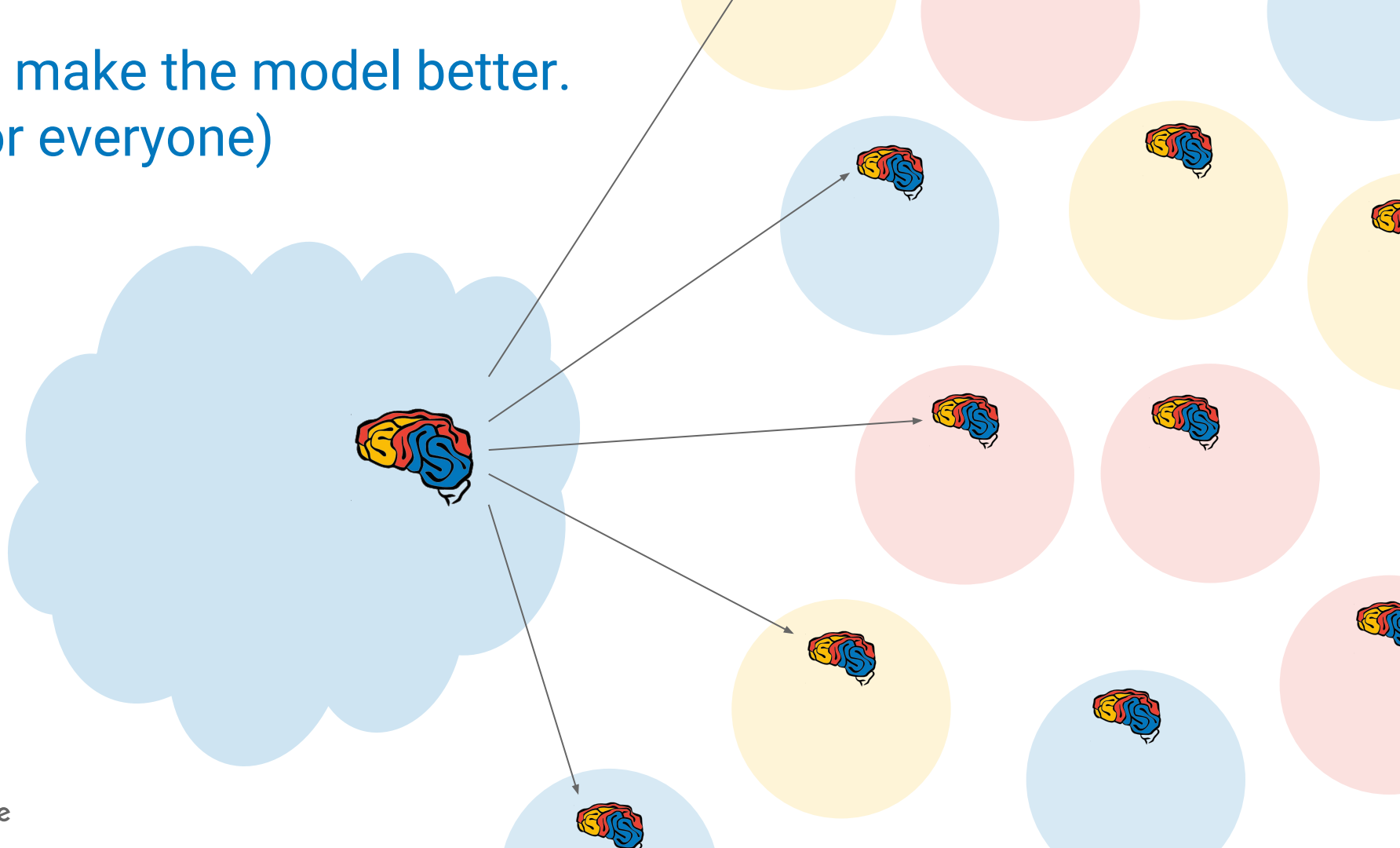
Federated Learning



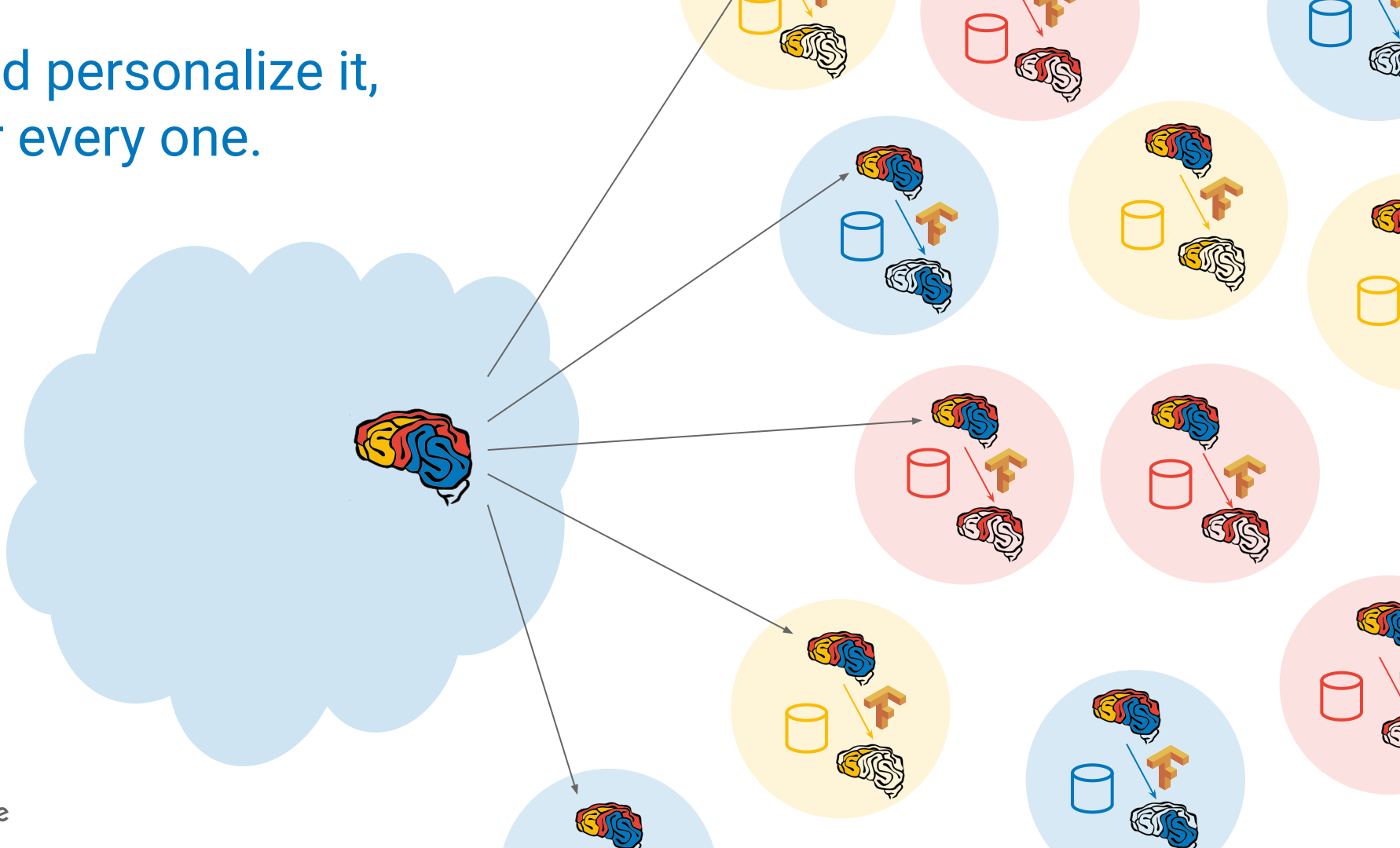
Federated Learning



To make the model better. (for everyone)



And personalize it, for every one.




Latency
Data Caps
Privacy
Offline
Power
Sensors

Latency

Data Caps

Privacy

In Vivo
Training & Evaluation

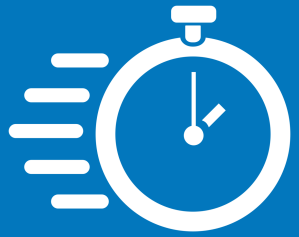
Offline

Power

Sensors

Personalization

Google



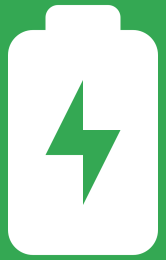
Latency

Data Caps

Privacy

In Vivo
Training & Evaluation

Offline



Power

Sensors

Personalization

Google

Applications of Federating Learning

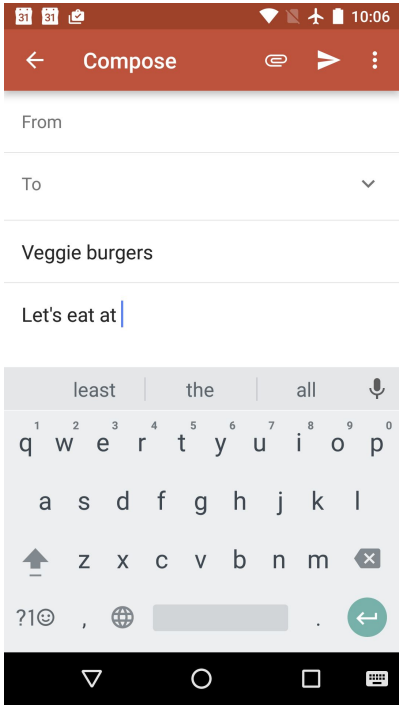
What makes a good application?

- On-device data is more relevant than server-side proxy data
- On-device data is privacy sensitive or large
- Labels can be inferred naturally from user interaction

Example applications

- Language modeling for mobile keyboards and voice recognition
- Image classification for predicting which photos people will share
- ...

Federated Learning in Gboard



Google Research Blog

The latest news from Research at Google

Federated Learning: Collaborative Machine Learning without Centralized Training Data

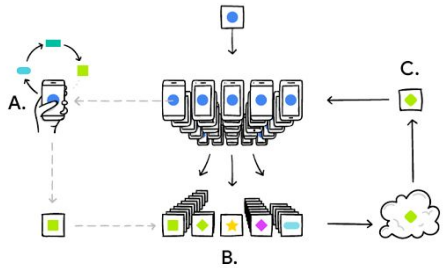
Thursday, April 06, 2017

Posted by Brendan McMahan and Daniel Ramage, Research Scientists

Standard machine learning approaches require centralizing the training data on one machine or in a datacenter. And Google has built one of the most secure and robust cloud infrastructures for processing this data to make our services better. Now for models trained from user interaction with mobile devices, we're introducing an additional approach: *Federated Learning*.

Federated Learning enables mobile phones to collaboratively learn a shared prediction model while keeping all the training data on device, decoupling the ability to do machine learning from the need to store the data in the cloud. This goes beyond the use of local models that make predictions on mobile devices (like the Mobile Vision API and On-Device Smart Reply) by bringing model *training* to the device as well.

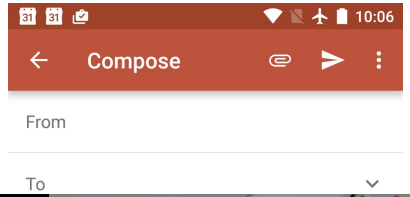
It works like this: your device downloads the current model, improves it by learning from data on your phone, and then summarizes the changes as a small focused update. Only this update to the model is sent to the cloud, using encrypted communication, where it is immediately averaged with other user updates to improve the shared model. All the training data remains on your device, and no individual updates are stored in the cloud.

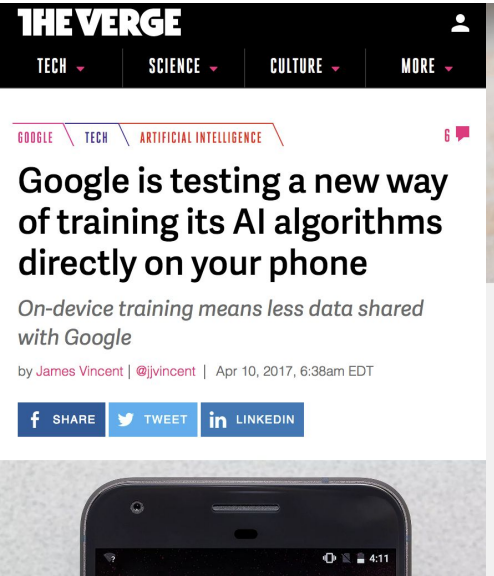


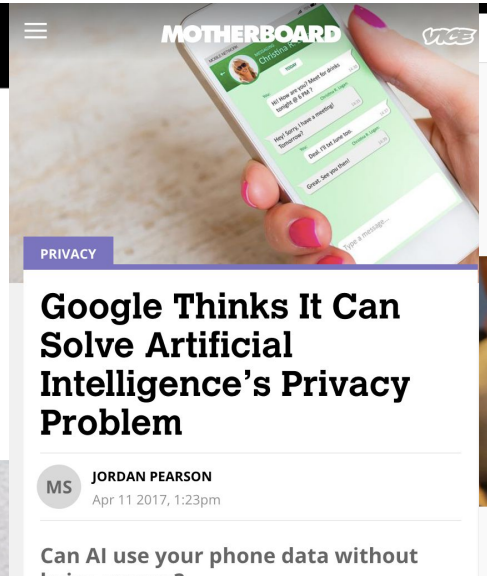
Your phone personalizes the model locally, based on your usage (A). Many users' updates are aggregated (B) to form a consensus change (C) to the shared model, after which the procedure is repeated.

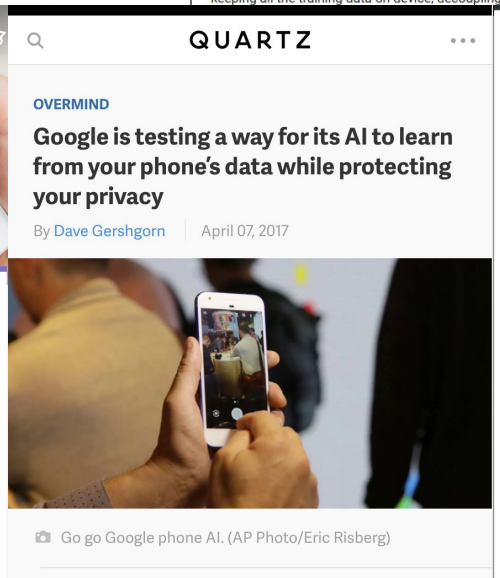
Google

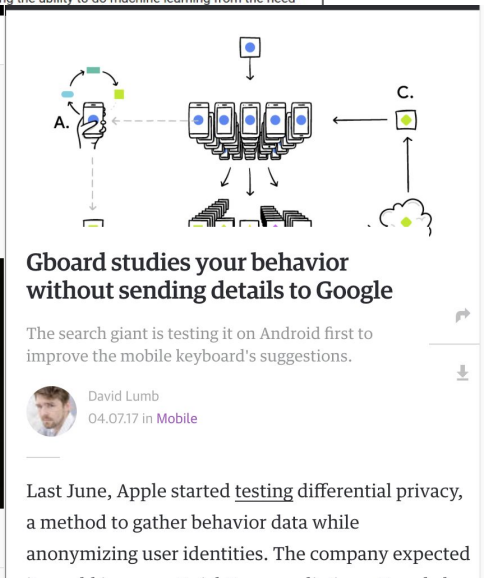
Federated Learning in Gboard











This Talk

1. Why Federated Learning?
2. **FL vs. other distributed optimization**
3. FL Algorithms
4. Minimizing Communication
5. Secure Aggregation
6. Differential Privacy

Atypical Federated Learning assumptions

Massively Distributed

Training data is stored across a very large number of devices

Atypical Federated Learning assumptions

Massively Distributed

Training data is stored across a very large number of devices

Limited Communication

Only a handful of rounds of unreliable communication with each devices

Atypical Federated Learning assumptions

Massively Distributed

Training data is stored across a very large number of devices

Limited Communication

Only a handful of rounds of unreliable communication with each devices

Unbalanced Data

Some devices have few examples, some have orders of magnitude more

Atypical Federated Learning assumptions

Massively Distributed

Training data is stored across a very large number of devices

Limited Communication

Only a handful of rounds of unreliable communication with each devices

Unbalanced Data

Some devices have few examples, some have orders of magnitude more

Highly Non-IID Data

Data on each device reflects one individual's usage pattern

Atypical Federated Learning assumptions

Massively Distributed

Training data is stored across a very large number of devices

Limited Communication

Only a handful of rounds of unreliable communication with each devices

Unbalanced Data

Some devices have few examples, some have orders of magnitude more

Highly Non-IID Data

Data on each device reflects one individual's usage pattern

Unreliable Compute Nodes

Devices go offline unexpectedly; expect faults and adversaries

Atypical Federated Learning assumptions

Massively Distributed

Training data is stored across a very large number of devices

Limited Communication

Only a handful of rounds of unreliable communication with each devices

Unbalanced Data

Some devices have few examples, some have orders of magnitude more

Highly Non-IID Data

Data on each device reflects one individual's usage pattern

Unreliable Compute Nodes

Devices go offline unexpectedly; expect faults and adversaries

Dynamic Data Availability

The subset of data available is non-constant, e.g. time-of-day vs. country

This Talk

1. Why Federated Learning?
2. FL vs. other distributed optimization
3. **FL Algorithms**
4. Minimizing Communication
5. Secure Aggregation
6. Differential Privacy

The Federated Averaging algorithm

Server

Until Converged:

1. Select a random subset (e.g. 1000) of the (online) clients

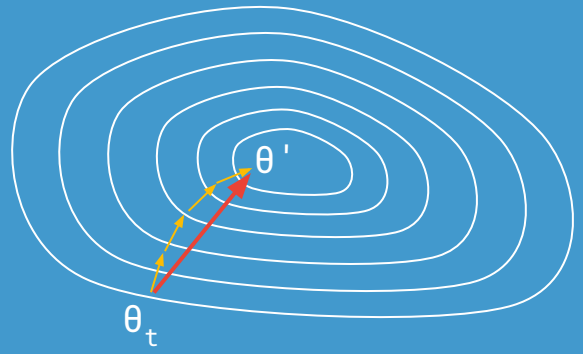
2. In parallel, send current parameters θ_t to those clients

Selected Client k

1. Receive θ_t from server.

2. Run some number of minibatch SGD steps, producing θ'

3. Return $\theta' - \theta_t$ to server.

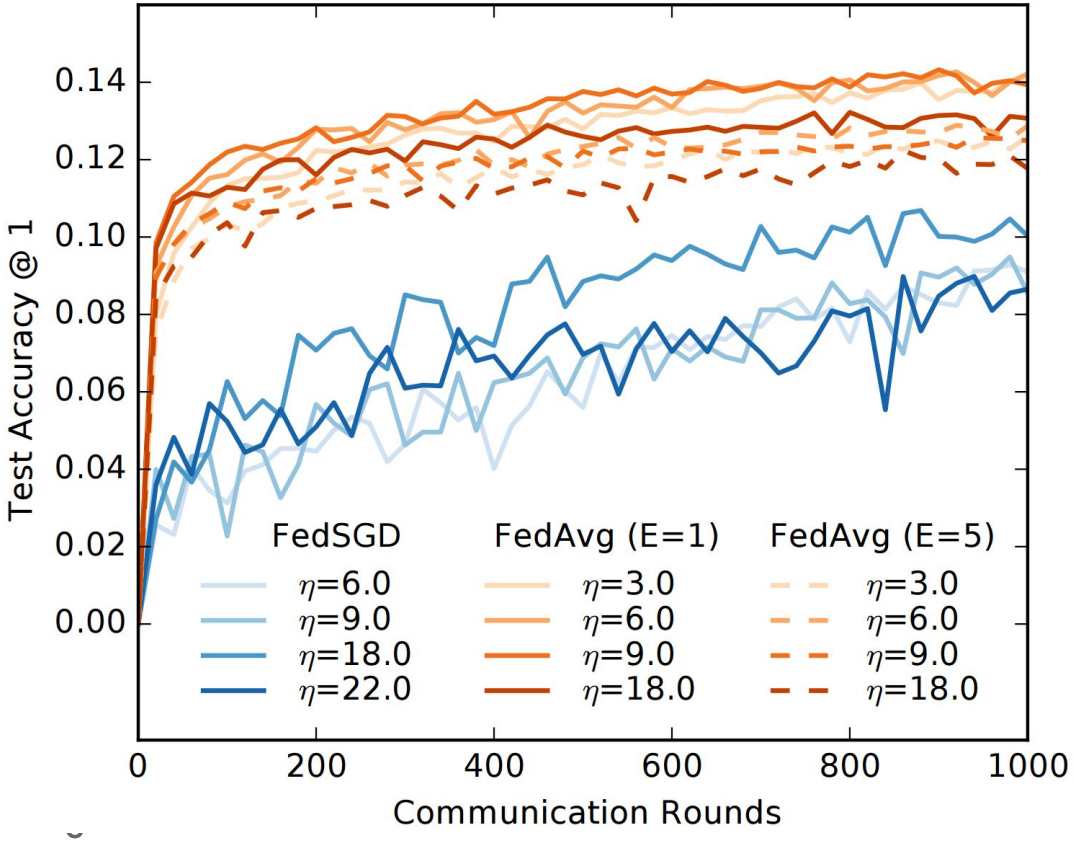


3. $\theta_{t+1} = \theta_t +$ data-weighted average of client updates

H. B. McMahan, *et al.* **Communication-Efficient Learning of Deep Networks from Decentralized Data.** AISTATS 2017

Large-scale LSTM for next-word prediction

Dataset: Large Social Network, 10m public posts, grouped by author.



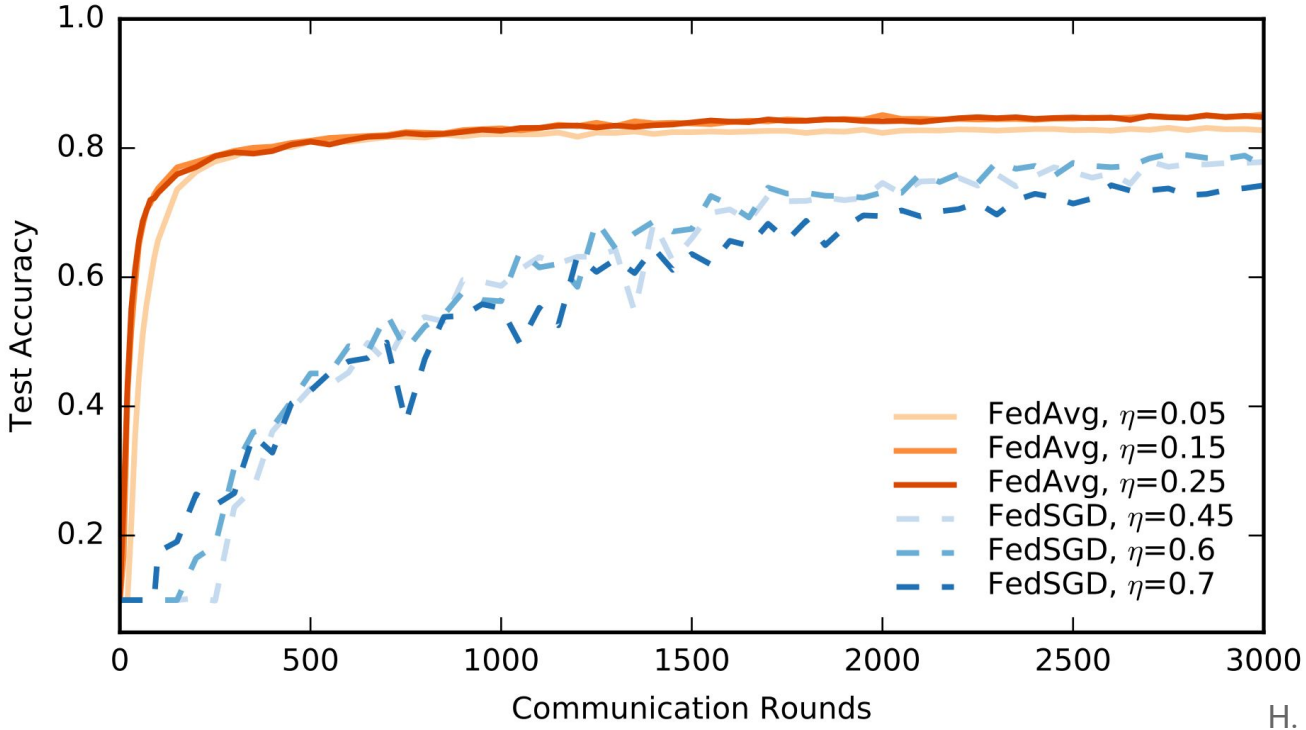
Rounds to reach 10.5% Accuracy

FedSGD	820
FedAvg	35

23x **decrease in communication rounds**

H. B. McMahan, *et al*.
Communication-Efficient Learning of Deep Networks from Decentralized Data. AISTATS 2017

CIFAR-10 convolutional model



Updates to reach 82%

SGD	31,000
FedSGD	6,600
FedAvg	630

49x decrease in communication (updates) vs SGD

(IID and balanced data)

H. B. McMahan, *et al*. **Communication-Efficient Learning of Deep Networks from Decentralized Data.** AISTATS 2017

Open Questions

Latency

Data Caps

Privacy

In Vivo

Training & Evaluation

Offline

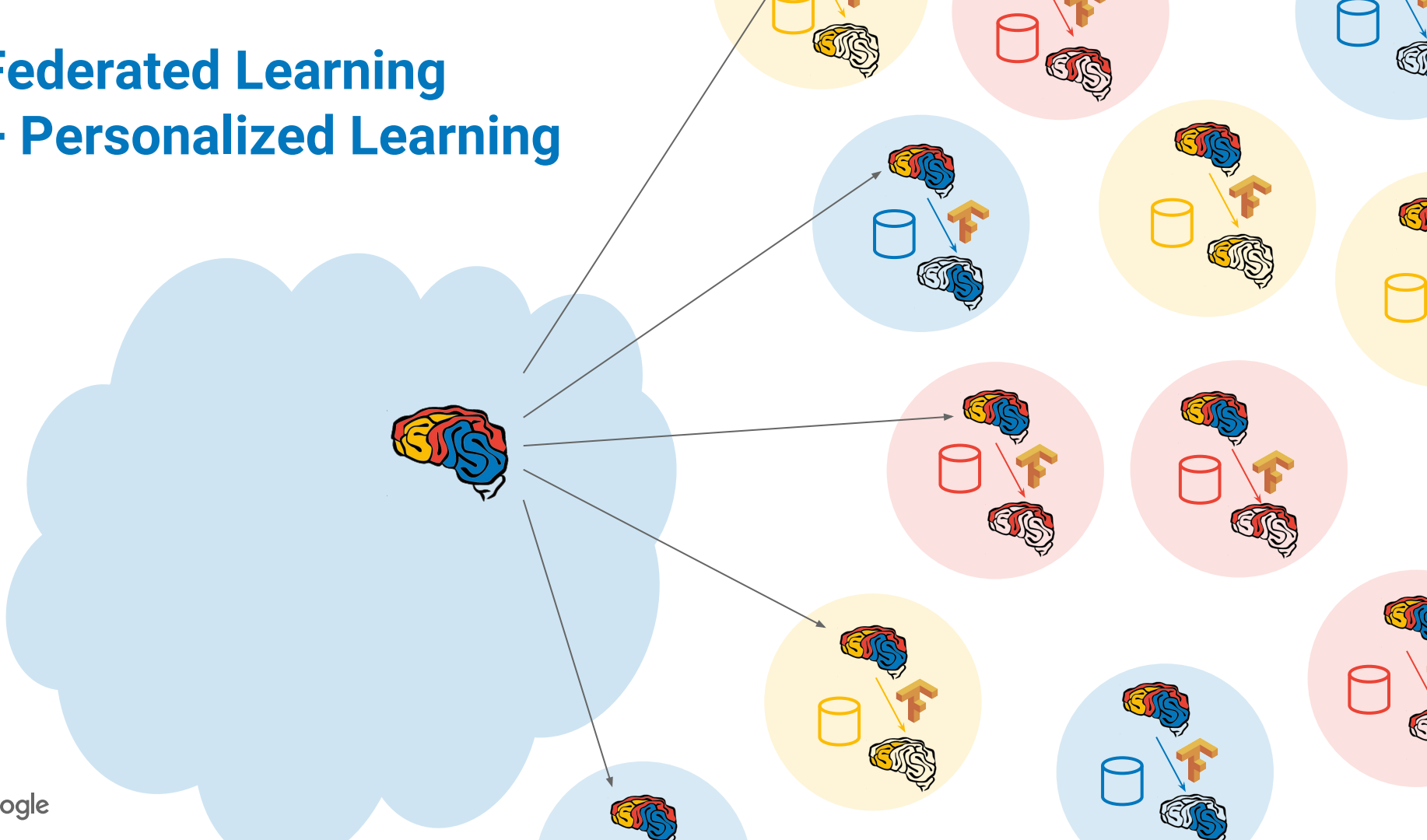
Power

Sensors

Personalization

Google

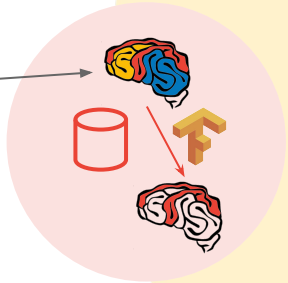
Federated Learning
+ Personalized Learning



Federated Learning
+ Personalized Learning

Take  as a ***recipe*** for 

(rather than a task-useful model on its own)

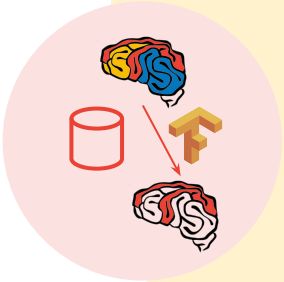


1. Multitask Learning

 represents a **regularizer** for training  , i.e. smoothing among models for similar users.

Take  as a ***recipe*** for

(rather than a task-useful model on its own)



P. Vanhaesebrouck, A. Bellet and M. Tommasi.
Decentralized Collaborative Learning of Personalized Models over Networks. AISTATS, 2017.

V. Smith, C.K. Chiang, M. Sanjabi, & A. Talwalkar
Federated Multi-Task Learning. arXiv preprint, 2017.

2. Learning to Learn

 represents a **learning procedure** that is biased towards learning  efficiently.

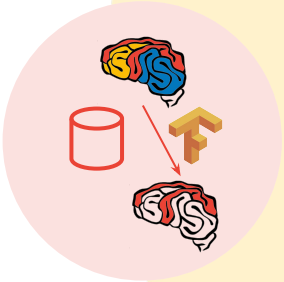
This procedure either replaces standard SGD or controls any free parameters (e.g. learning rate) when training  .

O. Wichrowska, N. Maheswaranathan, M. W. Hoffman, S. G. Colmenarejo, M. Denil, N. de Freitas, & J. Sohl-Dickstein. **Learned Optimizers that Scale and Generalize**. arXiv preprint, 2017.

N. Mishra, M. Rohaninejad, X. Chen, P. Abbeel. **Meta-Learning with Temporal Convolutions.** arXiv preprint, 2017.

Take  as a ***recipe*** for 

(rather than a task-useful model on its own)



Google

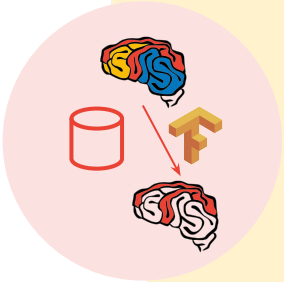
3. Model-Agnostic Meta Learning

 represents a good initialization for standard algorithms to very quickly learning  .

"Personalize to new users quickly" becomes the training objective for  by back-propagating through the training procedure for  .

Take  as a ***recipe*** for

(rather than a task-useful model on its own)



Chelsea Finn, P. Abbeel, S. Levine. **Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks.** arXiv preprint, 2017

Better algorithms than Federated Averaging?

Massively Distributed

Training data is stored across a very large number of devices

Limited Communication

Only a handful of rounds of unreliable communication with each devices

Unbalanced Data

Some devices have few examples, some have orders of magnitude more

Highly Non-IID Data

Data on each device reflects one individual's usage pattern

Unreliable Compute Nodes

Devices go offline unexpectedly; expect faults and adversaries

Dynamic Data Availability

The subset of data available is non-constant, e.g. time-of-day vs. country

This Talk

1. Why Federated Learning?
2. FL vs. other distributed optimization
3. FL Algorithms
4. **Minimizing Communication**
5. Secure Aggregation
6. Differential Privacy

The Federated Averaging algorithm

Server

Until Converged:

1. Select a random subset (e.g. 1000) of the (online) clients

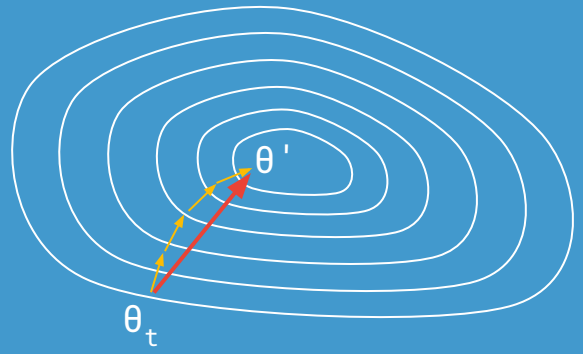
2. In parallel, send current parameters θ_t to those clients

Selected Client k

1. Receive θ_t from server.

2. Run some number of minibatch SGD steps, producing θ'

3. Return $\theta' - \theta_t$ to server.



3. $\theta_{t+1} = \theta_t +$ data-weighted average of client updates

H. B. McMahan, *et al.* **Communication-Efficient Learning of Deep Networks from Decentralized Data.** AISTATS 2017

The Federated Averaging algorithm

Server

Until Converged:

1. Select a random subset (e.g. 1000) of the (online) clients

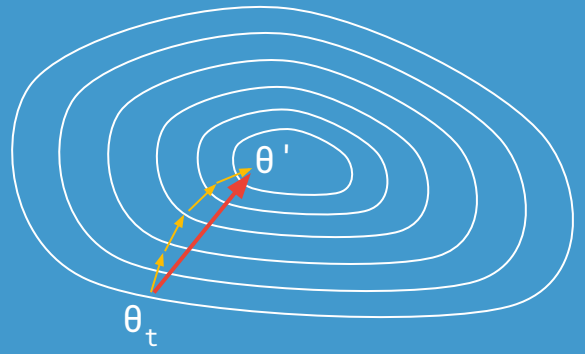
2. In parallel, send current parameters θ_t to those clients

Selected Client k

1. Receive θ_t from server.

2. Run some number of minibatch SGD steps, producing θ'

3. Return $\theta' - \theta_t$ to server. **Potential bottleneck**



3. $\theta_{t+1} = \theta_t$ + data-weighted average of client updates

H. B. McMahan, *et al.* **Communication-Efficient Learning of Deep Networks from Decentralized Data.** AISTATS 2017

Upload time can be bottleneck

Asymmetric connection speed

Download generally faster than upload

`http://www.speedtest.net/reports/`

Upload time can be bottleneck

Asymmetric connection speed

Download generally faster than upload

`http://www.speedtest.net/reports/`

Some markets more data sensitive

E.g. India, Nigeria, Brazil, ...

Upload time can be bottleneck

Asymmetric connection speed

Download generally faster than upload

`http://www.speedtest.net/reports/`

Some markets more data sensitive

E.g. India, Nigeria, Brazil, ...

Secure Aggregation

Further increases the data needed to communicate

Upload time can be bottleneck

Asymmetric connection speed

Download generally faster than upload

`http://www.speedtest.net/reports/`

Some markets more data sensitive

E.g. India, Nigeria, Brazil, ...

Secure Aggregation

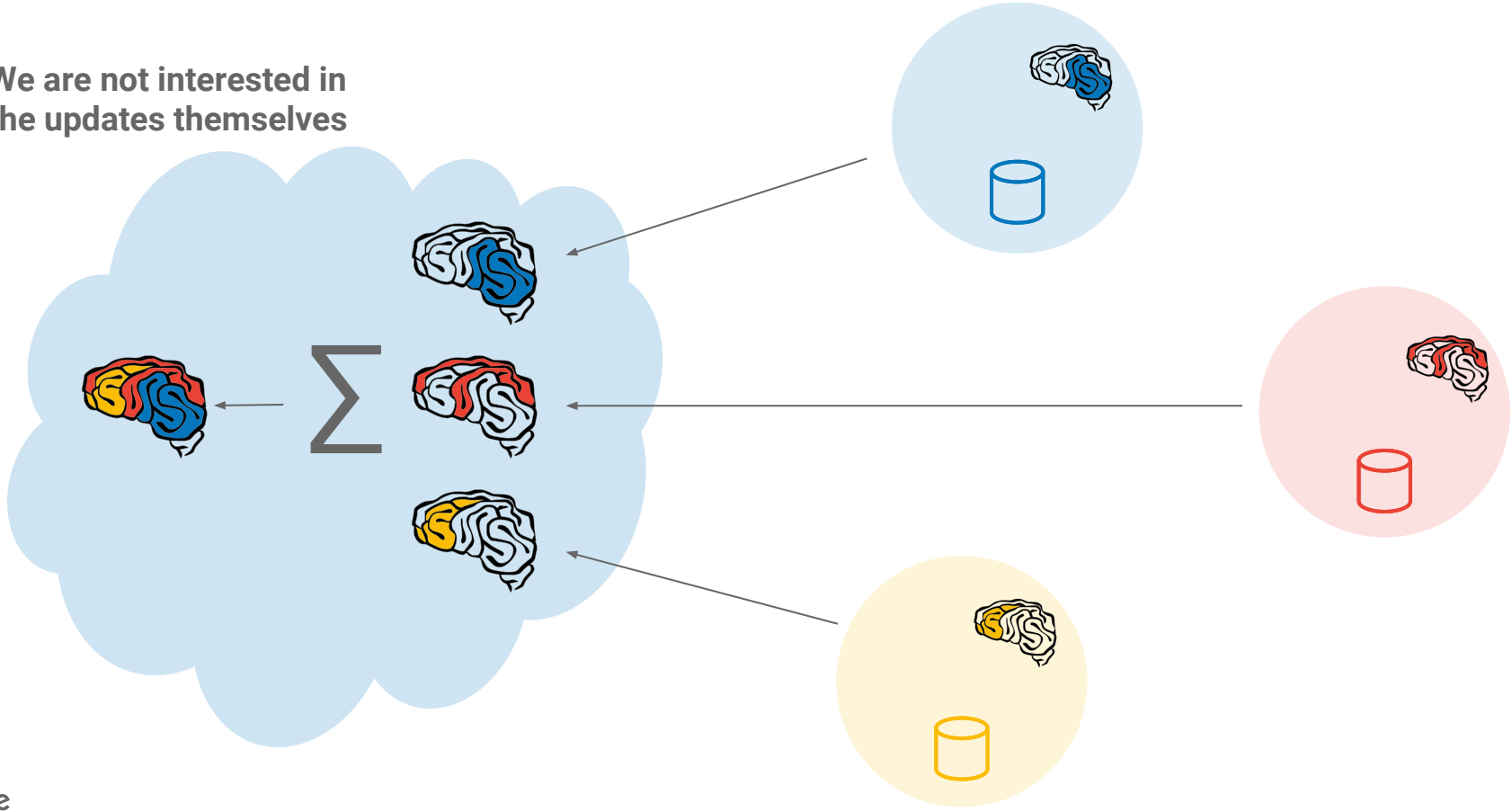
Further increases the data needed to communicate

Energy consumption

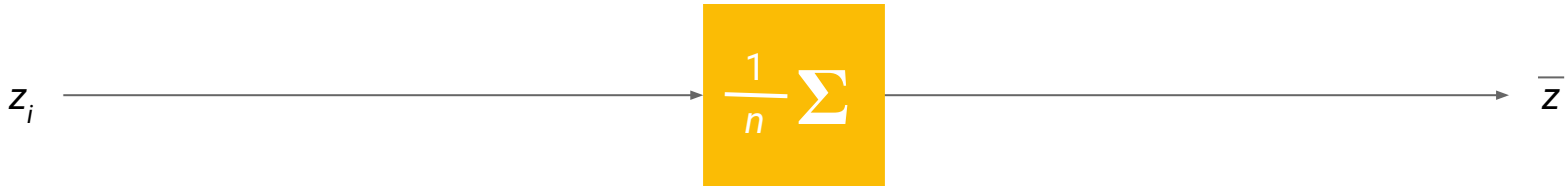
Data transmission generally power intensive

Federated Learning

We are not interested in
the updates themselves



Distributed Mean Estimation with Limited Communication



Distributed Mean Estimation with Limited Communication

A. T. Suresh, F.. Yu, S. Kumar, & H. B. McMahan. **Distributed Mean Estimation with Limited Communication**. ICML 2017.

Distributed Mean Estimation with Limited Communication

Stochastic Binary Quantization
(1 bit per dimension)

$$Y_i(j) = \begin{cases} X_i^{\max} & \text{w.p. } \frac{X_i(j)-X_i^{\min}}{X_i^{\max}-X_i^{\min}} \\ X_i^{\min} & \text{otherwise.} \end{cases}$$

A. T. Suresh, F.. Yu, S. Kumar, & H. B. McMahan. **Distributed Mean Estimation with Limited Communication**. ICML 2017.

Distributed Mean Estimation with Limited Communication

Stochastic Binary Quantization
(1 bit per dimension)

Mean Squared Error: $\mathcal{E}(\pi_{sb}, X^n) = \Theta\left(\frac{d}{n} \cdot \frac{1}{n}\sum_{i=1}^{n} ||X_i||_2^2\right)$

If d is large, this is prohibitive.

A. T. Suresh, F.. Yu, S. Kumar, & H. B. McMahan. **Distributed Mean Estimation with Limited Communication**. ICML 2017.

Distributed Mean Estimation with Limited Communication

A. T. Suresh, F.. Yu, S. Kumar, & H. B. McMahan. **Distributed Mean Estimation with Limited Communication**. ICML 2017.

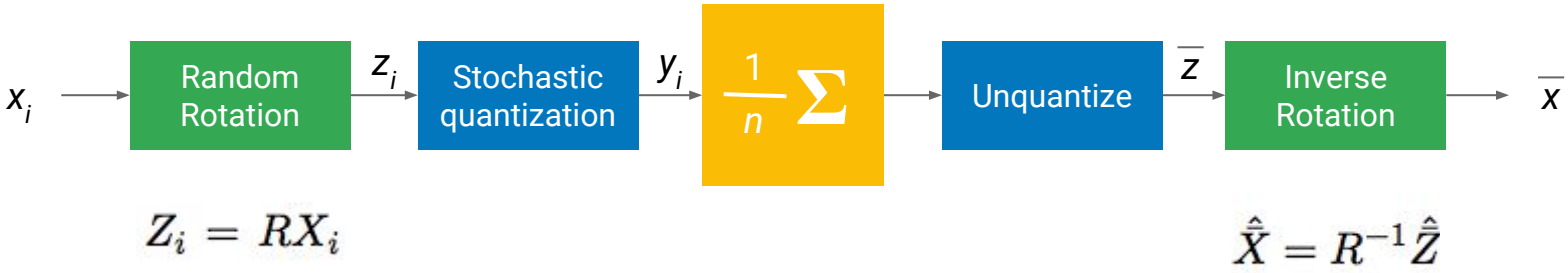
Distributed Mean Estimation with Limited Communication

$$\mathcal{E}(\pi_{srb}, X^n) = O\left(\frac{\log d}{n} \cdot \frac{1}{n}\sum_{i=1}^{n} ||X_i||_2^2\right)$$

Much better for large d
Can be modified to O(1/n)

A. T. Suresh, F.. Yu, S. Kumar, & H. B. McMahan. **Distributed Mean Estimation with Limited Communication**. ICML 2017.

Distributed Mean Estimation with Limited Communication



Efficiency:

Avoid representing, transmitting, or inverting R, which is $d \times d$.

A. T. Suresh, F.. Yu, S. Kumar, & H. B. McMahan. **Distributed Mean Estimation with Limited Communication**. ICML 2017.

Distributed Mean Estimation with Limited Communication

Structured Matrix: R = HD

D: diagonal matrix of i.i.d. Rademacher entries (±1)
H: Walsh-Hadamard matrix

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad \text{recurse: } \begin{bmatrix} H & H \\ H & -H \end{bmatrix} \qquad \text{inverse: } HH^{\mathrm{T}} = nI_n$$

A. T. Suresh, F.. Yu, S. Kumar, & H. B. McMahan. **Distributed Mean Estimation with Limited Communication**. ICML 2017.

Distributed Mean Estimation with Limited Communication

$$Z_i = RX_i$$

$$\hat{\hat{X}} = R^{-1}\hat{\hat{Z}}$$

Structured Matrix: R = HD

$$\mathcal{E}(\pi_{srb}, X^n) = O\left(\frac{\log d}{n} \cdot \frac{1}{n}\sum_{i=1}^{n} ||X_i||_2^2\right)$$

Rotation & Inverse Rotation

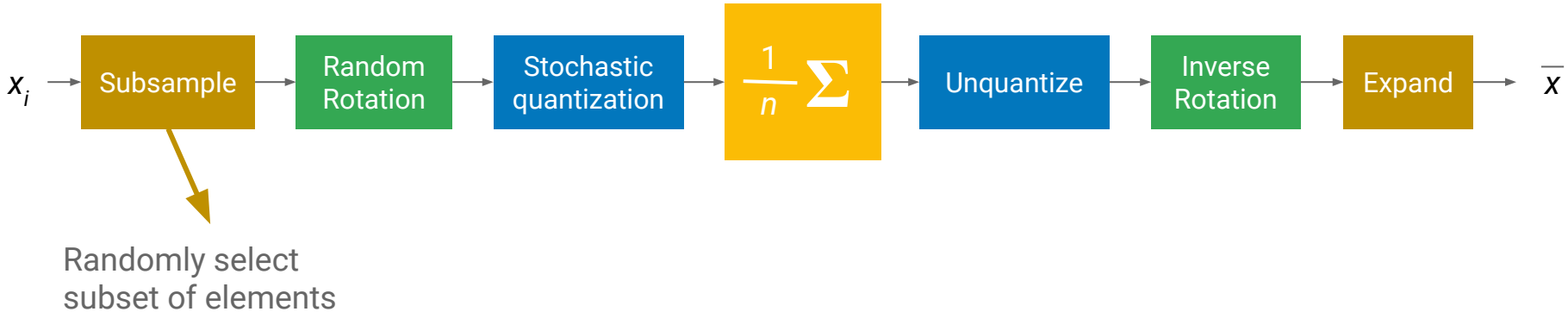
Time: O(d log d)

Additional Space: O(1)

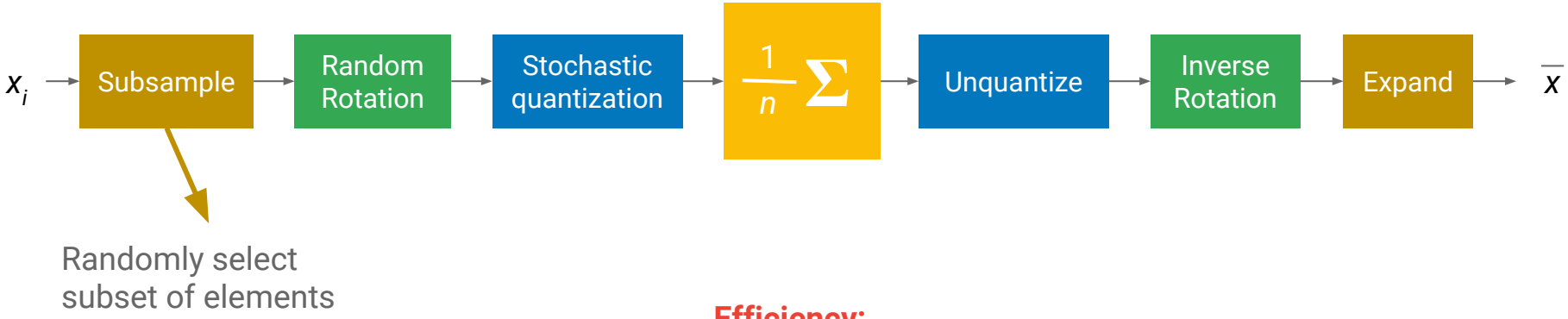
Communication (seed): O(1)

A. T. Suresh, F.. Yu, S. Kumar, & H. B. McMahan. **Distributed Mean Estimation with Limited Communication**. ICML 2017.

Distributed Mean Estimation with Limited Communication



Distributed Mean Estimation with Limited Communication

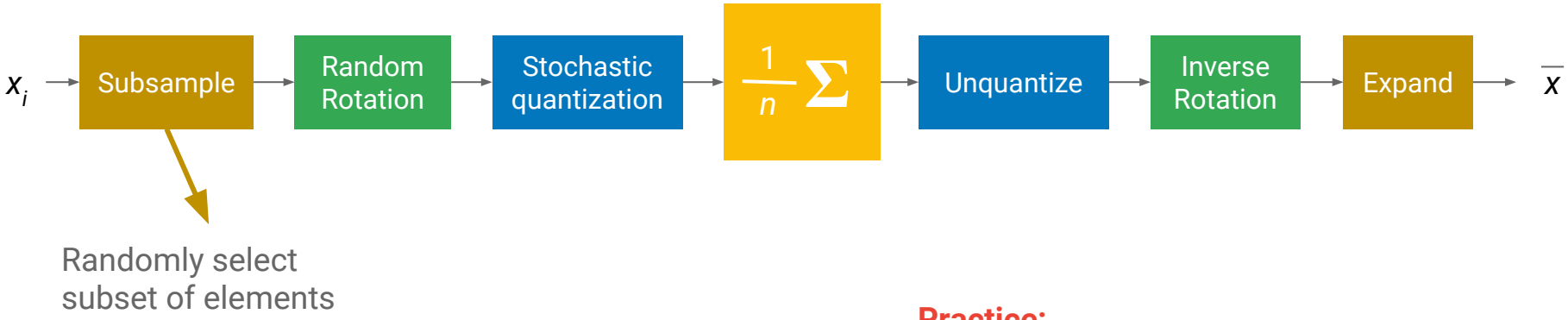


Efficiency:

Communicate only subsampled values

Corresponding indices can be represented as a random seed

Distributed Mean Estimation with Limited Communication



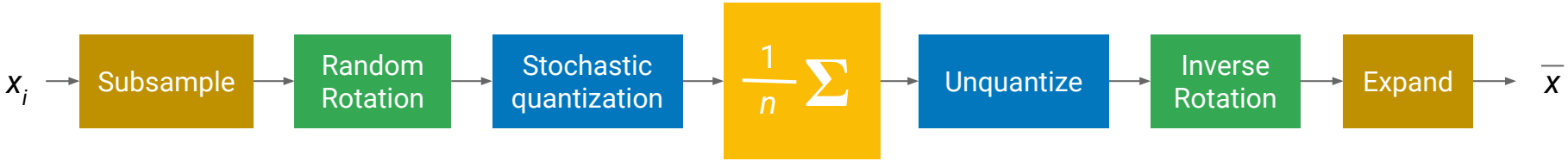
Practice:

Some subsampling and moderate quantization

Better than

No subsampling and aggressive quantization

Distributed Mean Estimation with Limited Communication



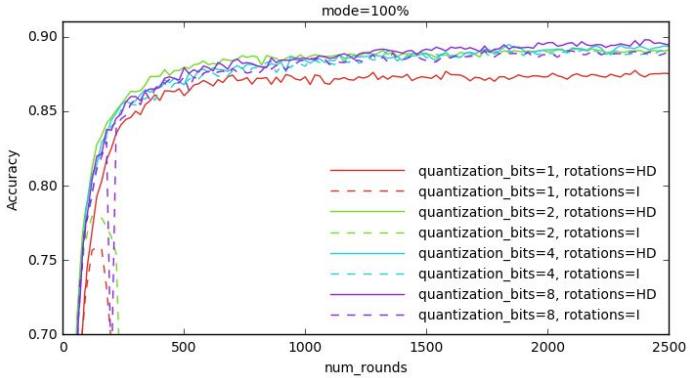
For instance:
D. Alistarh, D. Grubic, J. Li, R. Tomioka, M. Vojnovic. **QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding**. NIPS 2017

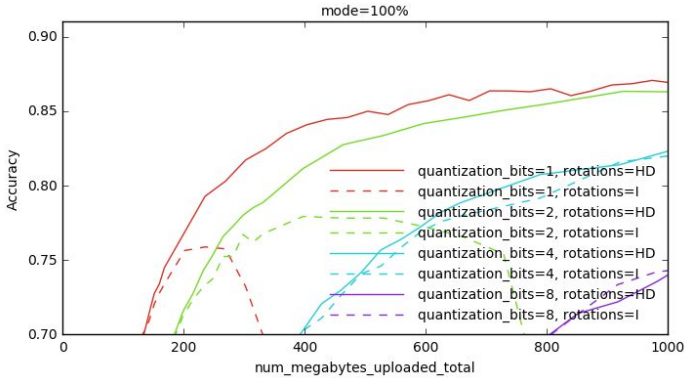
W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, H. Li. **TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning**. NIPS 2017

Existing related work

- Mostly alternative to **Quantization**
- Complementary to **Subsampling** and **Rotation**
  - Not necessarily efficient (MPI GPU-to-GPU communication)

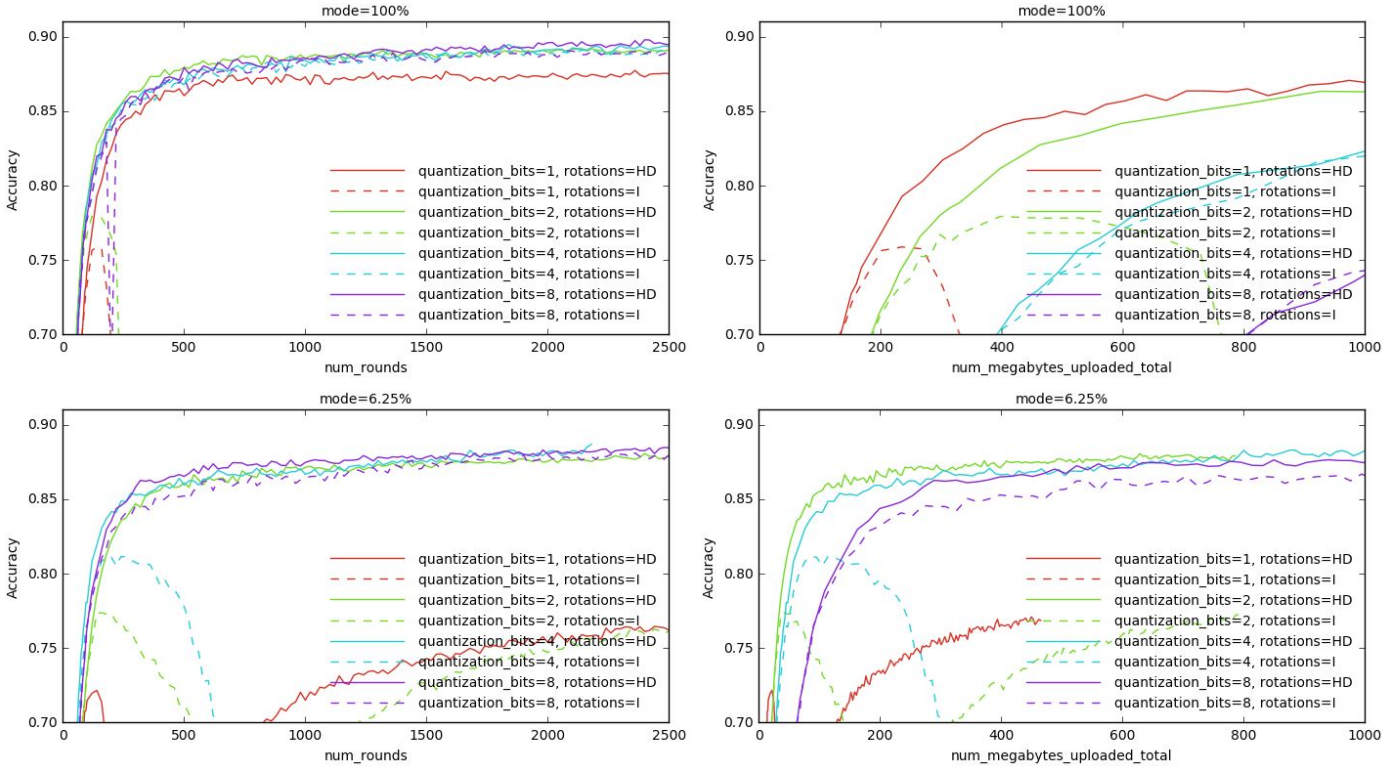
Federated Learning with Compressed Updates





J. Konečný, H. B. McMahan, F. Yu, P. Richtarik, A. T. Suresh, D. Bacon **Federated Learning: Strategies for Improving Communication Efficiency**. arXiv:1610.05492

Federated Learning with Compressed Updates



J. Konečný, H. B. McMahan, F. Yu, P. Richtarik, A. T. Suresh, D. Bacon **Federated Learning: Strategies for Improving Communication Efficiency**. arXiv:1610.05492

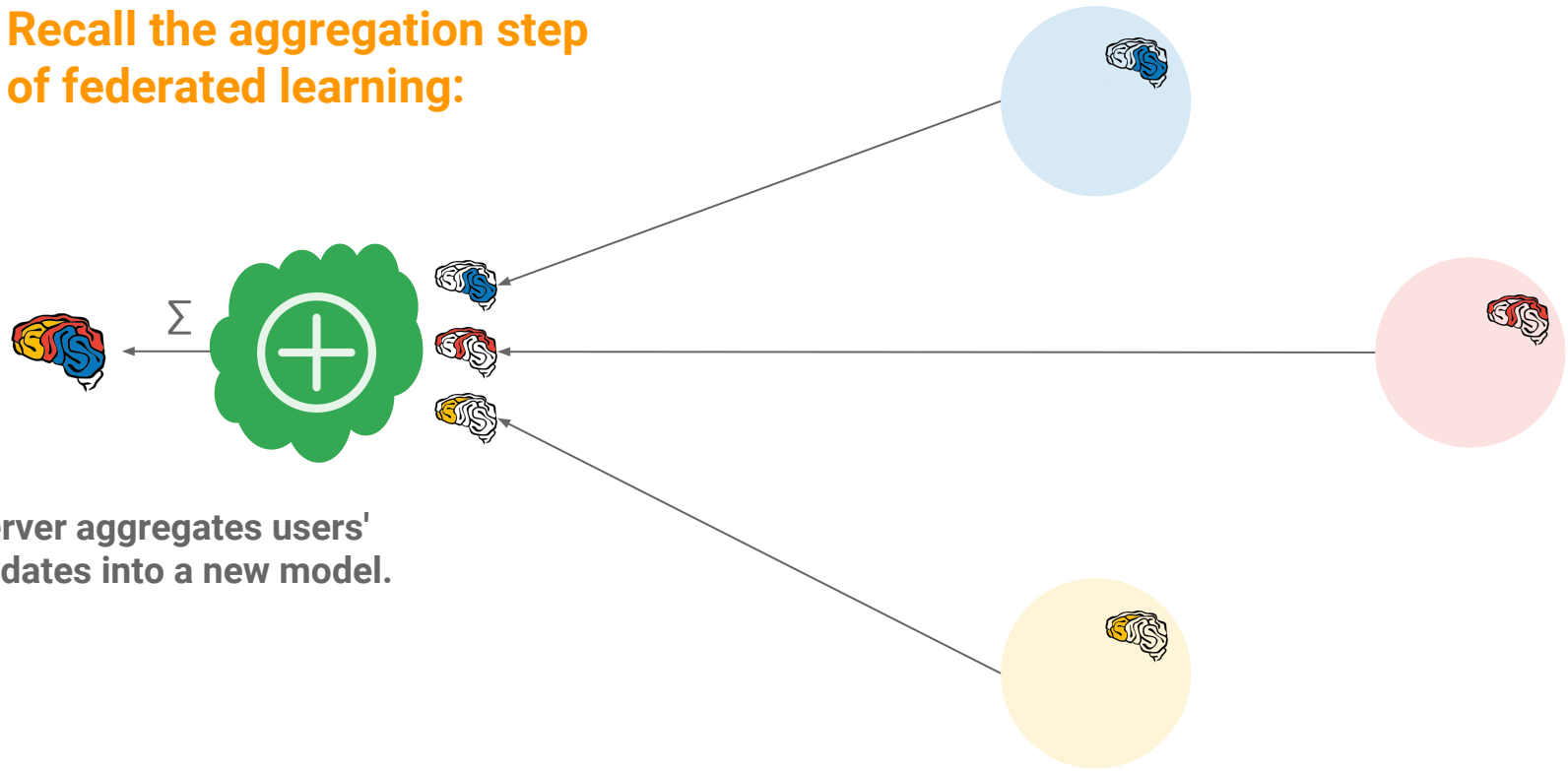
Google

This Talk

1. Why Federated Learning?
2. FL vs. other distributed optimization
3. FL Algorithms
4. Minimizing Communication
5. **Secure Aggregation**
6. Differential Privacy

Federated Learning

Recall the aggregation step of federated learning:



Server aggregates users' updates into a new model.

Federated Learning

Might these updates contain privacy-sensitive data?

Federated Learning

1. Ephemeral

Might these updates contain privacy-sensitive data?

Federated Learning

1. Ephemeral
2. **Focused**

Might these updates contain privacy-sensitive data?

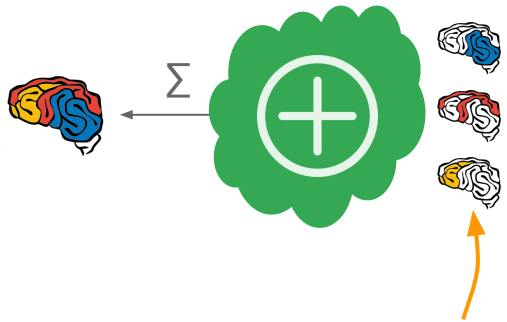
Federated Learning

Might these updates contain privacy-sensitive data?

1. Ephemeral
2. Focused

→ *Federated Learning is a privacy-preserving technology.*

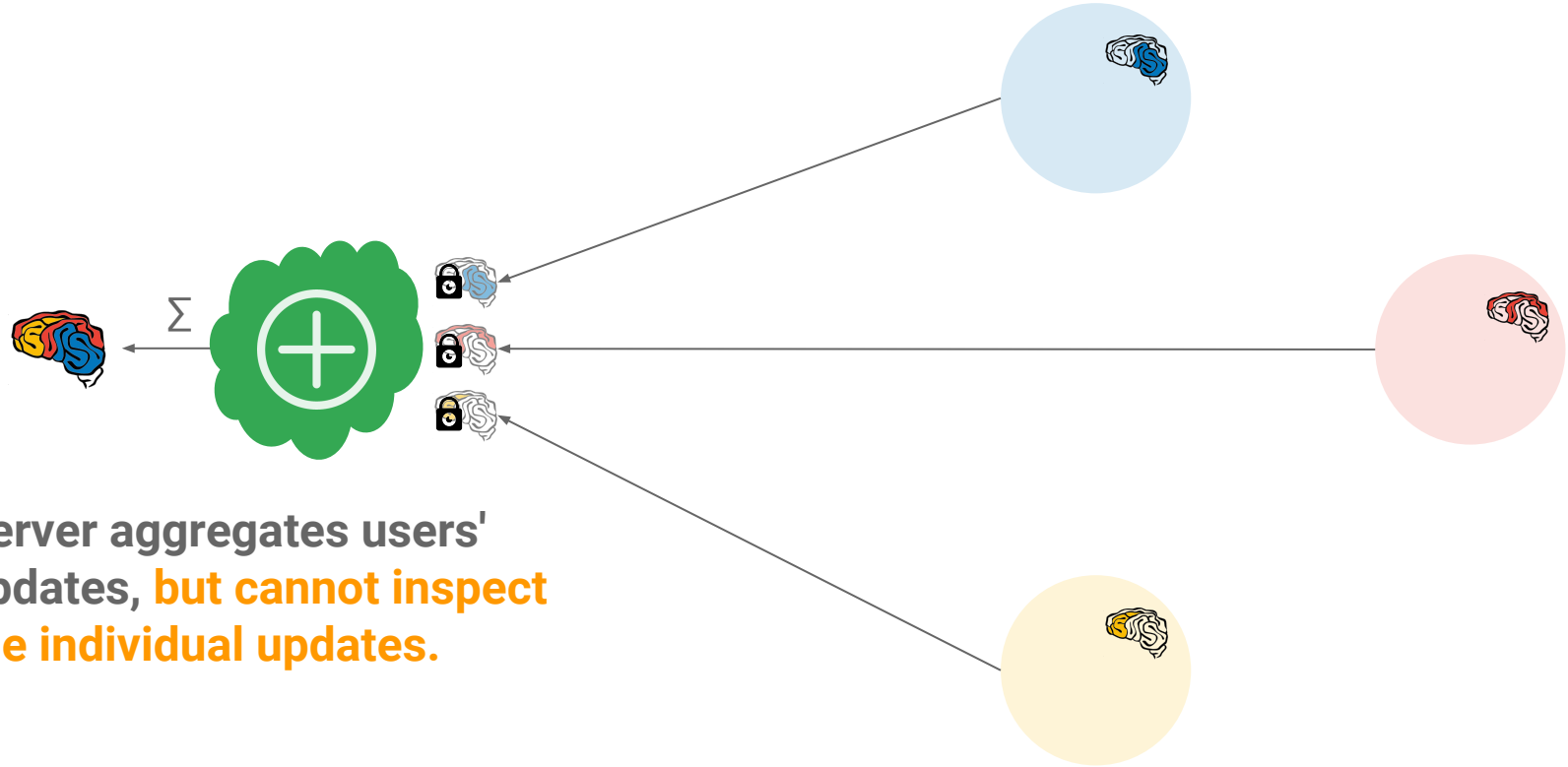
Federated Learning



Might these updates contain privacy-sensitive data?

1. Ephemeral
2. Focused
3. **Only in Aggregate**

Wouldn't it be great if...



Server aggregates users' updates, but cannot inspect the individual updates.

Secure Aggregation.

Secure Aggregation.

Existing protocols either:

Transmit
a *lot* of data

Fail when
users drop out

(or both)

A novel protocol for Secure Aggregation.

K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, K. Seth. **Practical Secure Aggregation for Privacy-Preserving Machine Learning.** *To appear at CCS 2017.*

Existing protocols either:

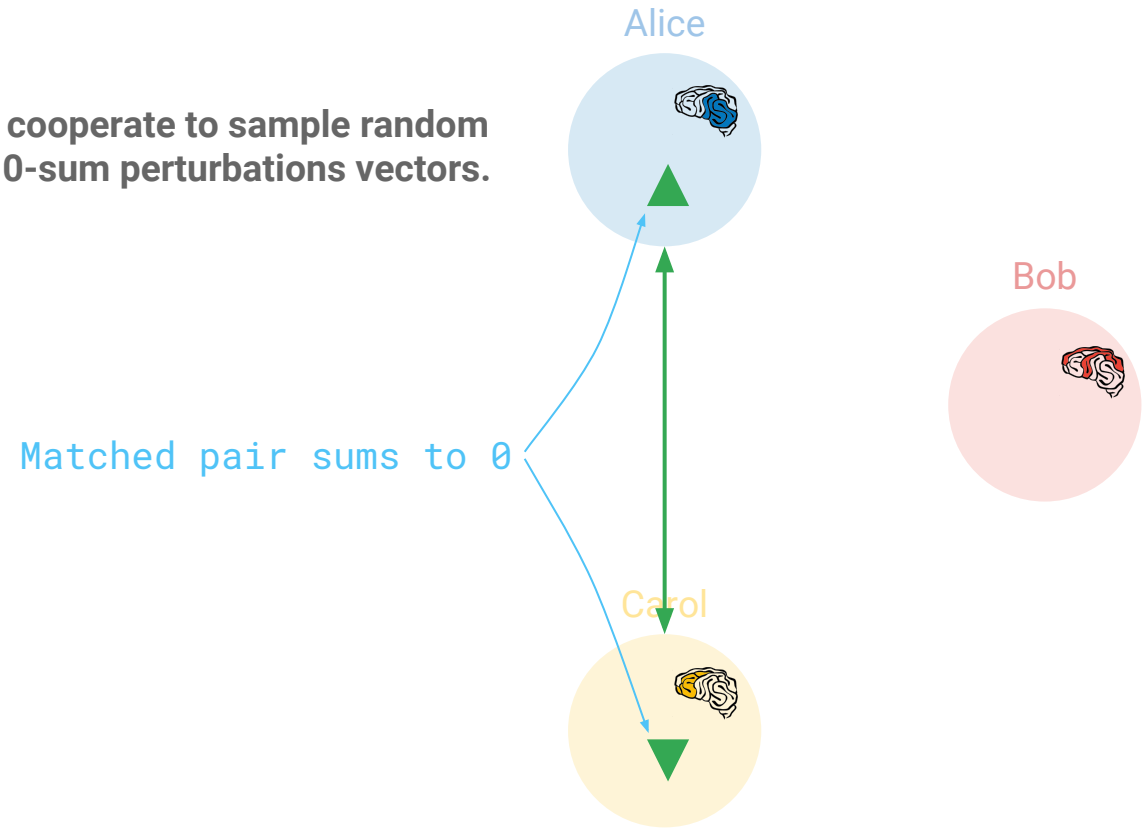
Transmit
a *lot* of data

Fail when
users drop out

(or both)

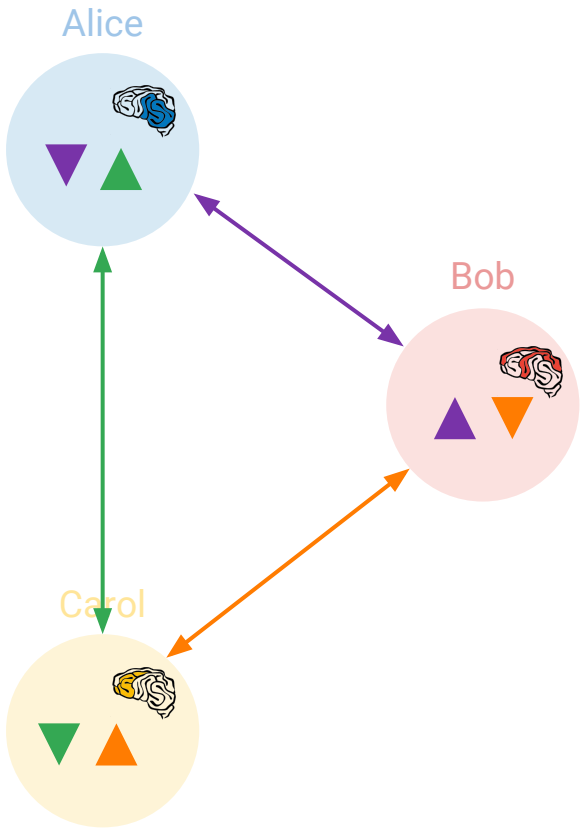
Random positive/negative pairs, *aka* antiparticles

Devices cooperate to sample random pairs of 0-sum perturbations vectors.

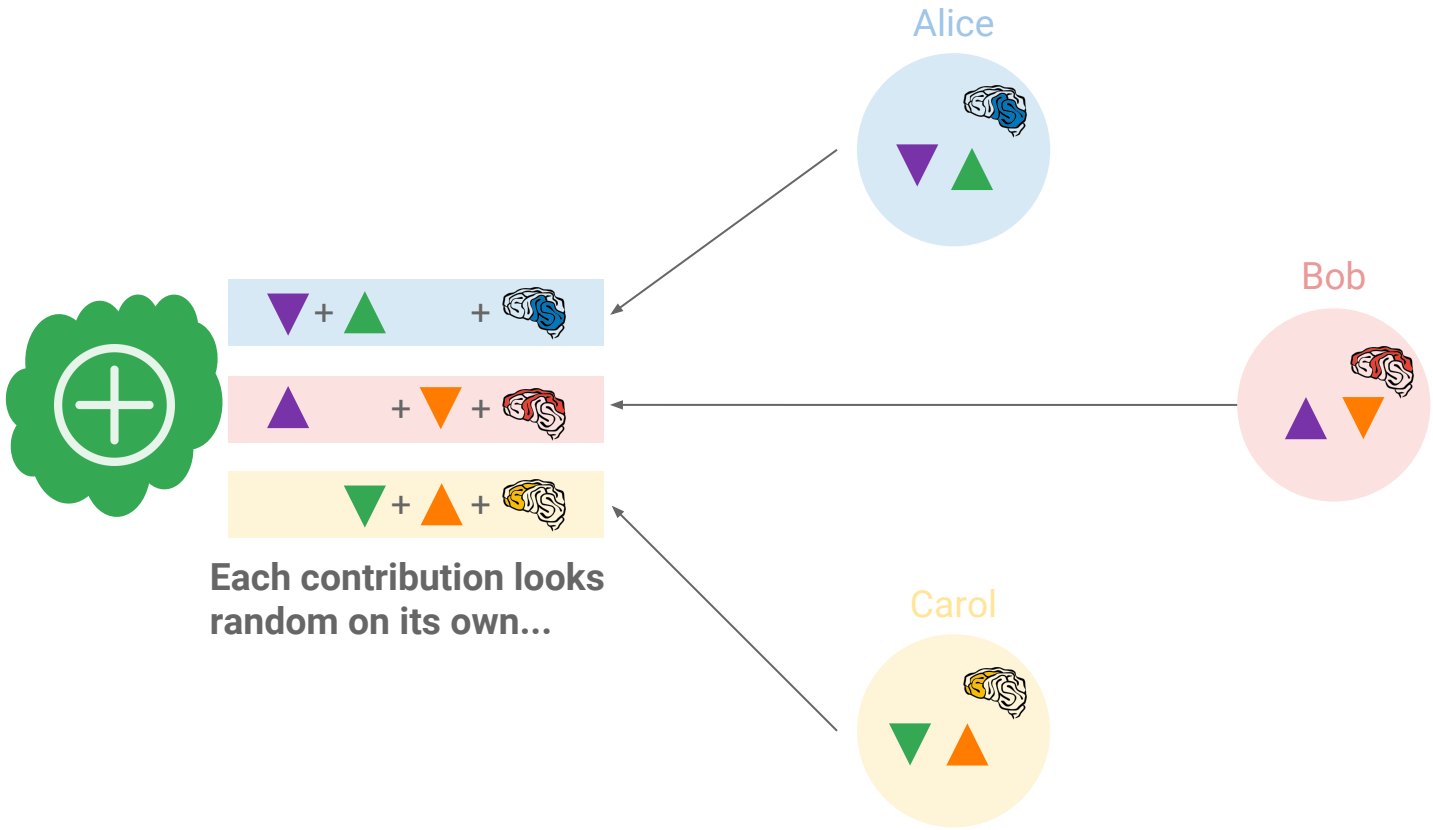


Random positive/negative pairs, *aka* antiparticles

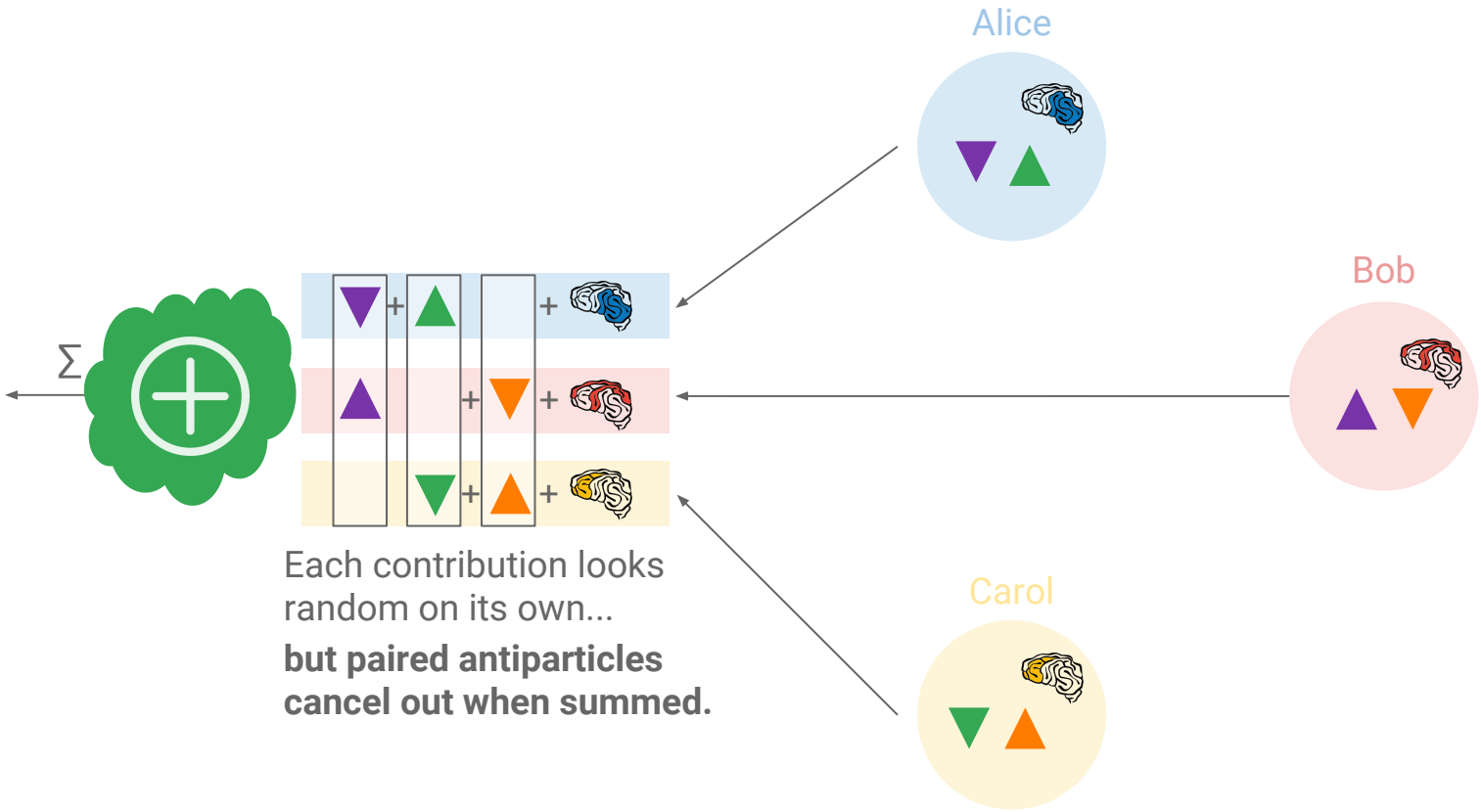
Devices cooperate to sample random pairs of 0-sum perturbations vectors.



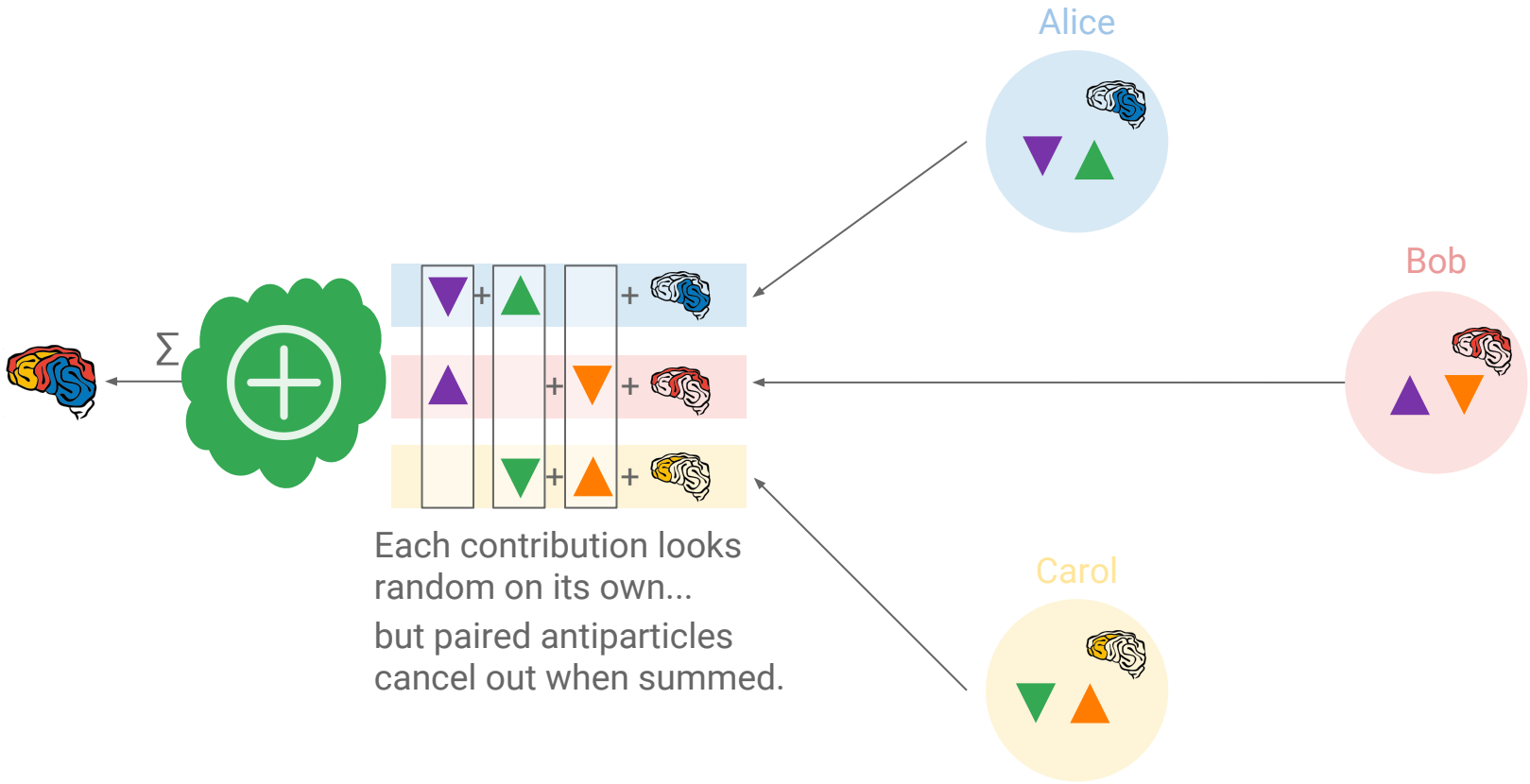
Add antiparticles before sending to the server



The antiparticles cancel when summing contributions

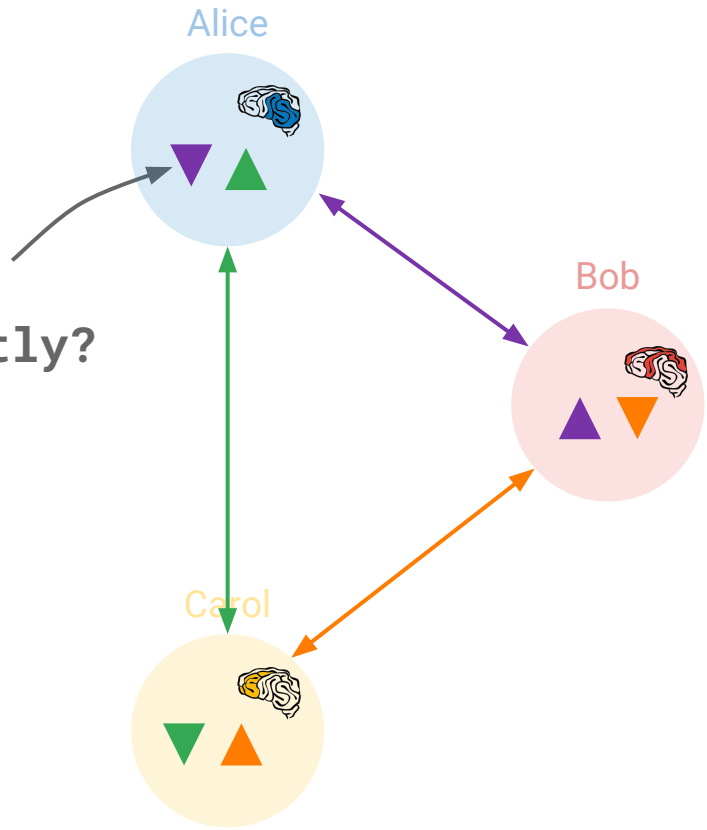


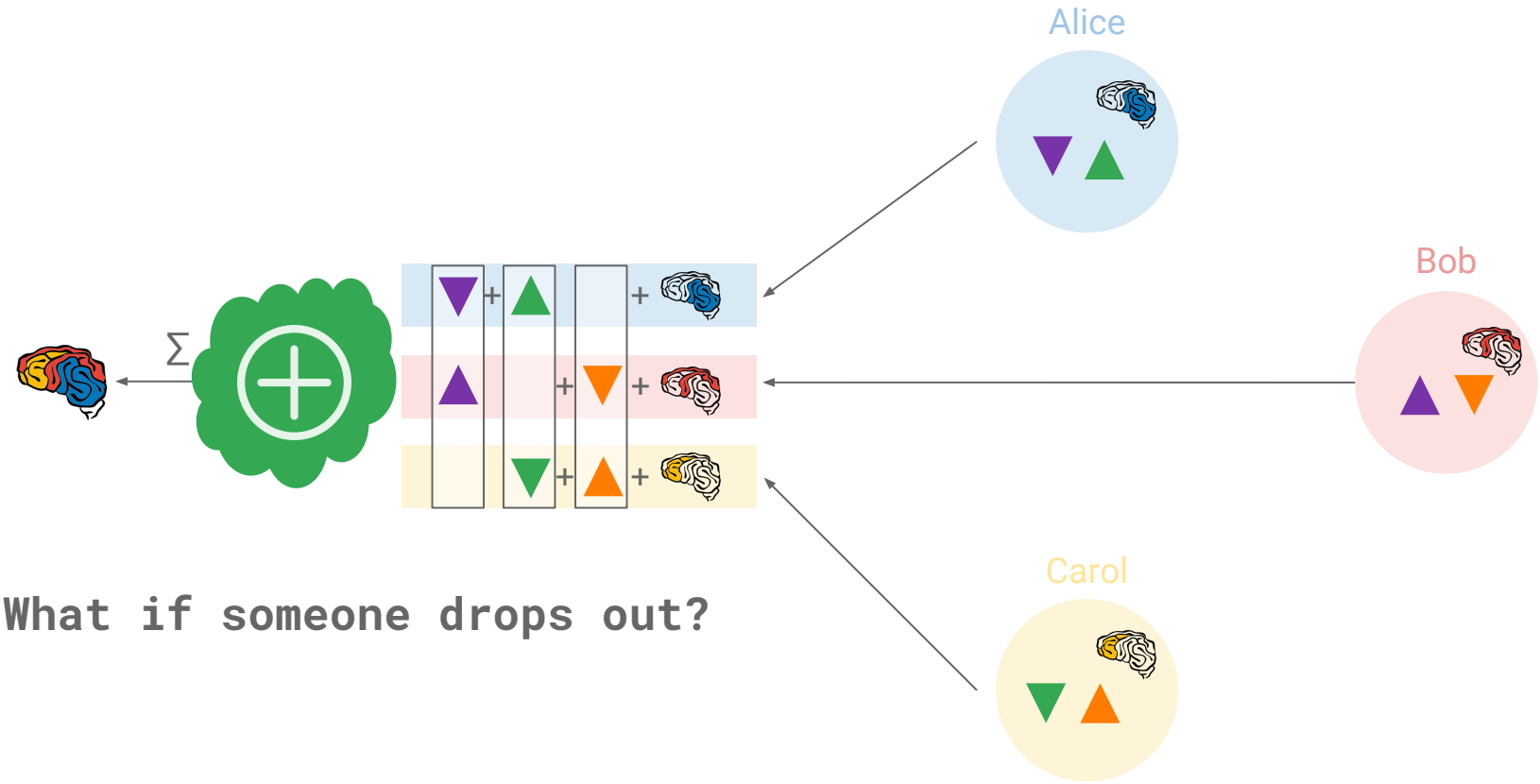
Revealing the sum.



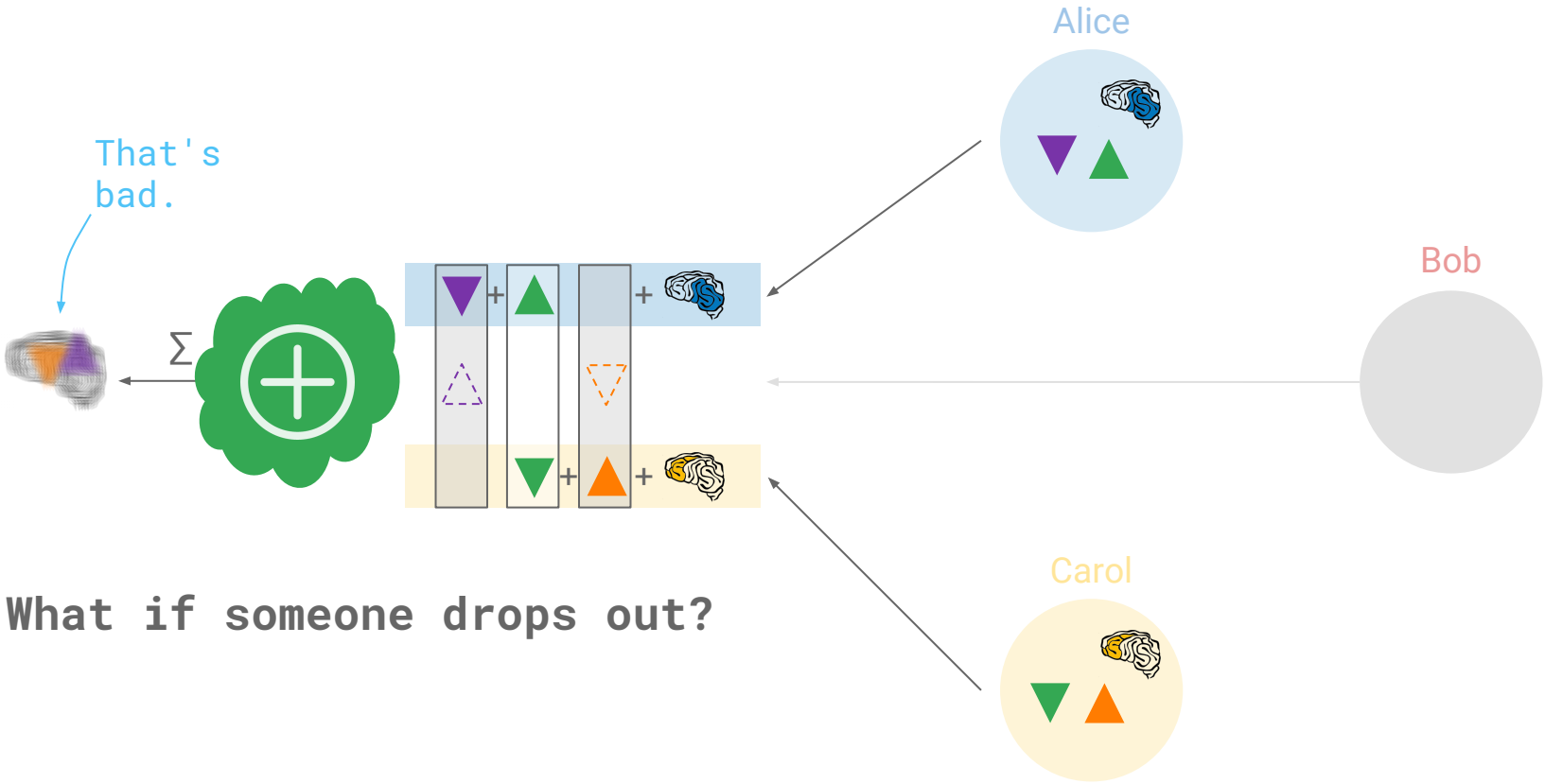
But there are two problems...

1. These vectors are big!
   How do users agree efficiently?



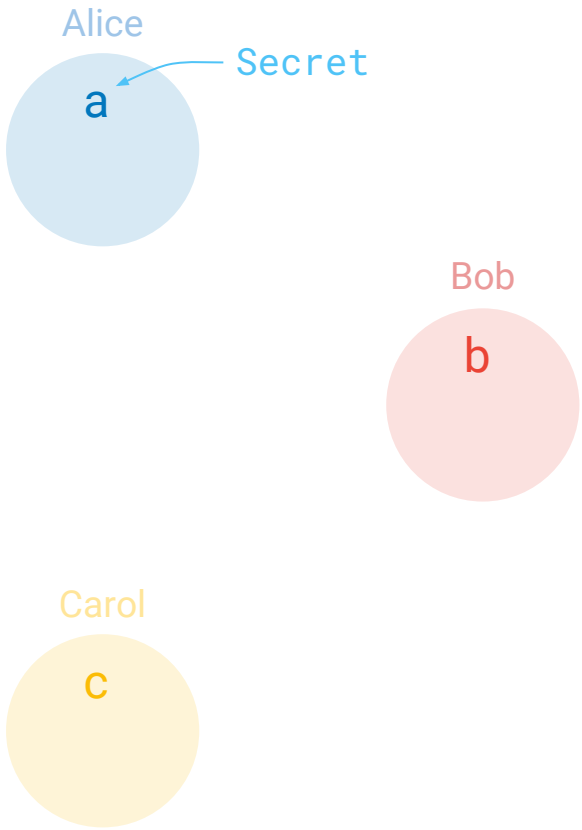


2. What if someone drops out?

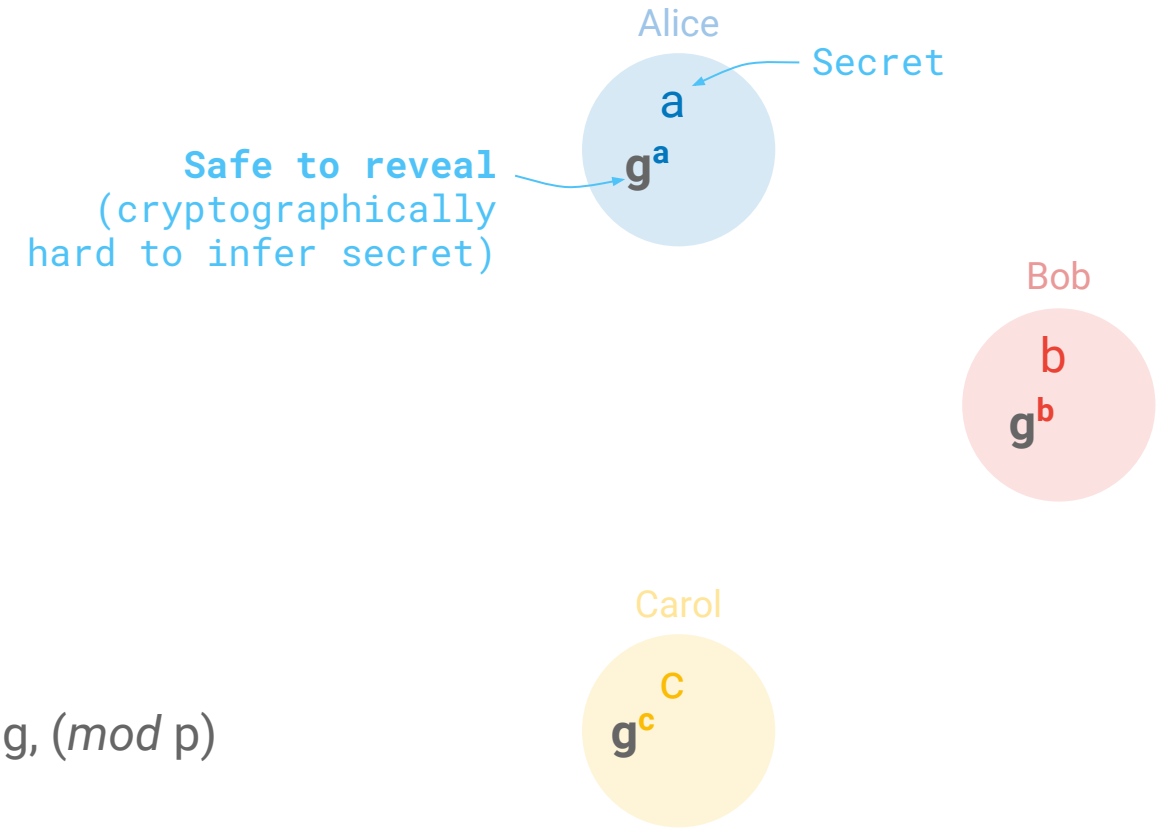


2. What if someone drops out?

Pairwise Diffie-Hellman Key Agreement

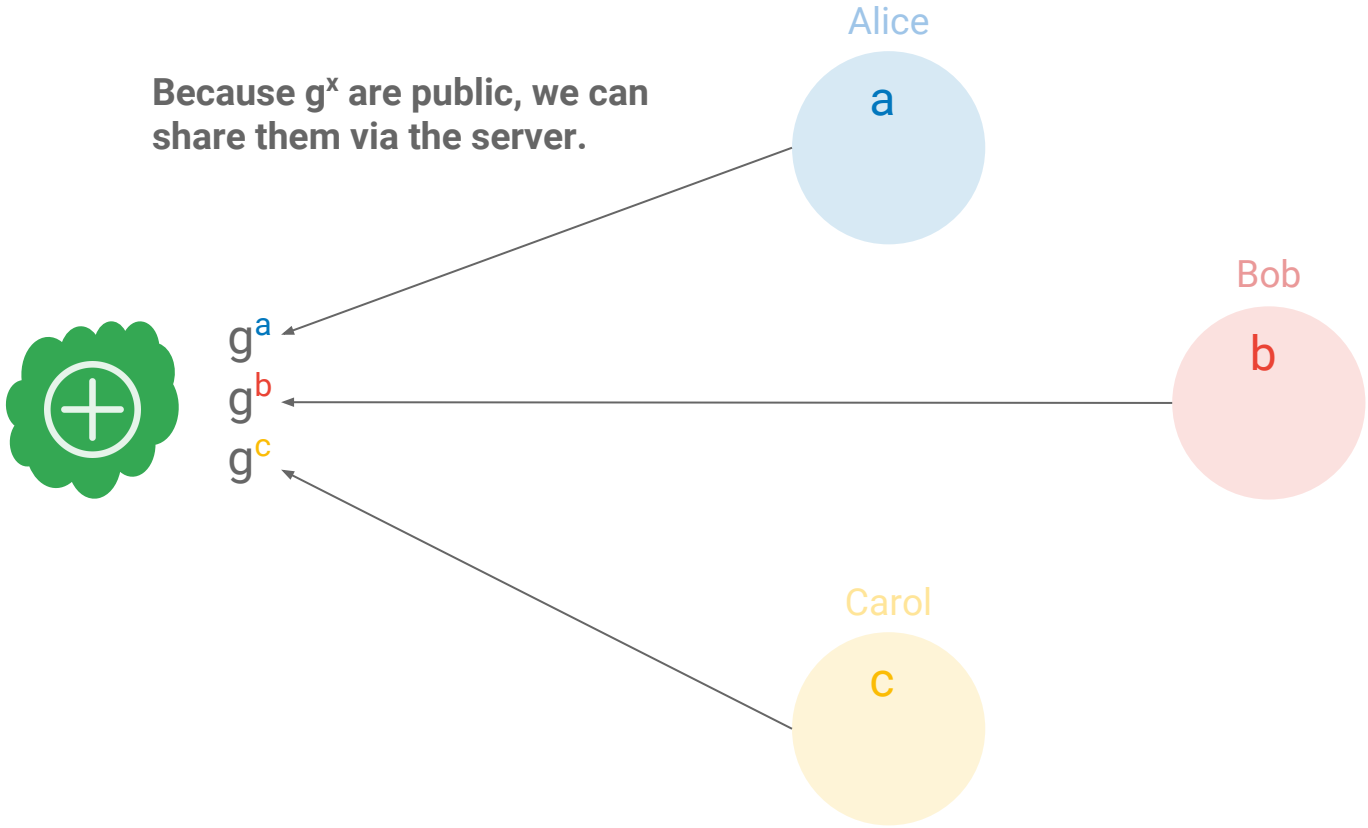


Pairwise Diffie-Hellman Key Agreement

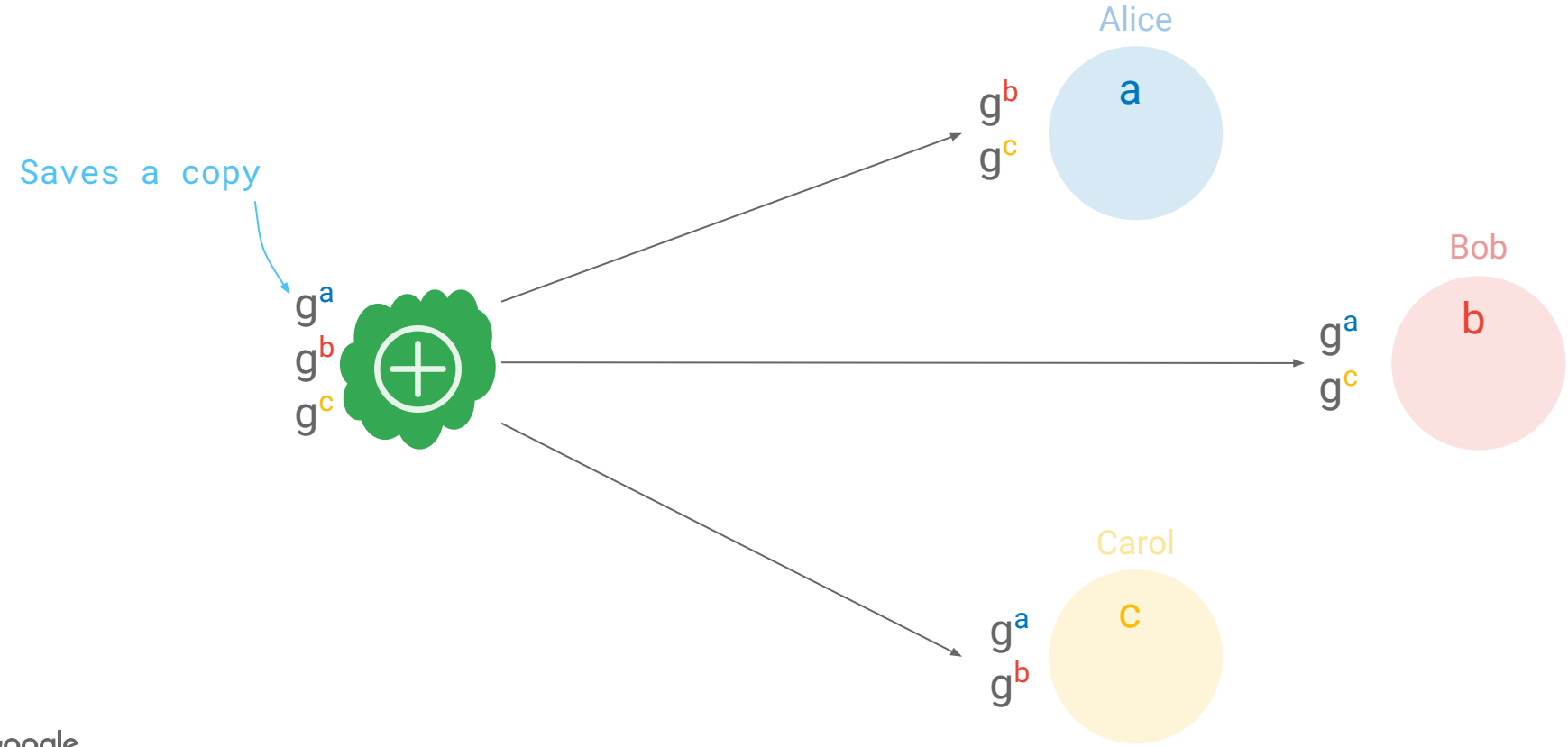


Public parameters: g, (*mod* p)

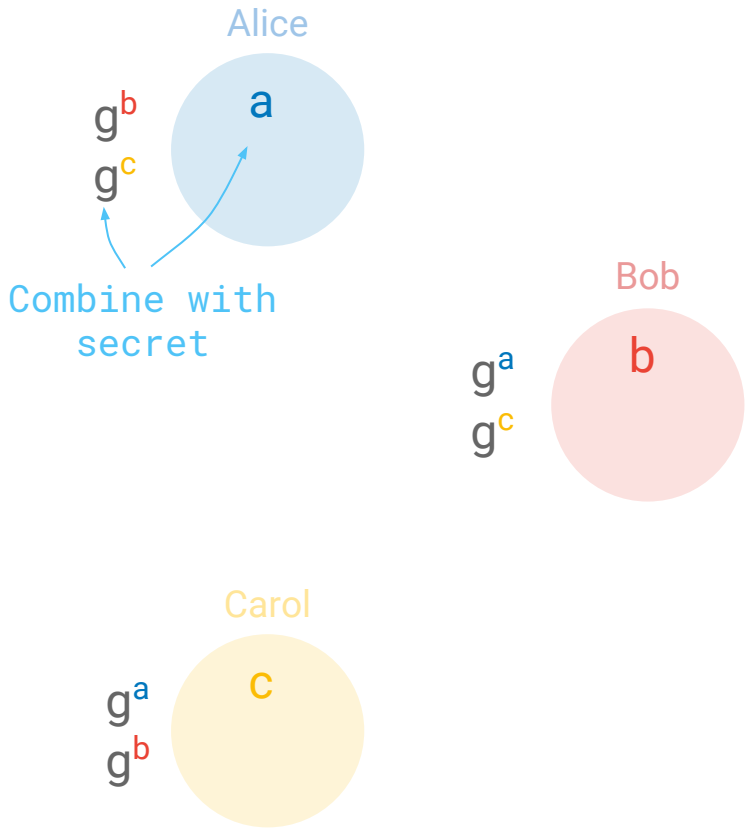
Pairwise Diffie-Hellman Key Agreement



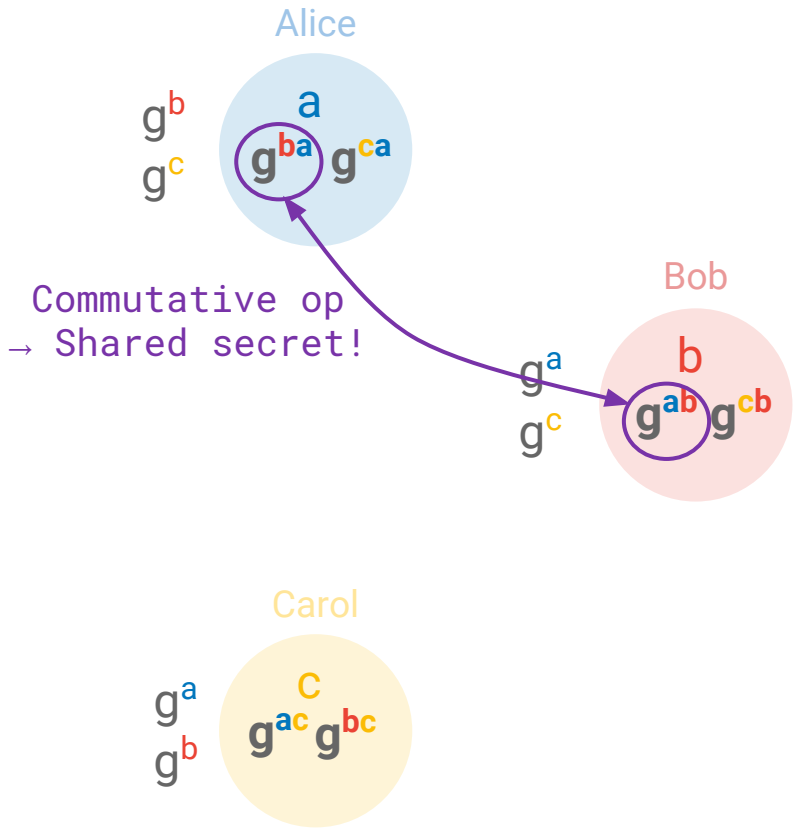
Pairwise Diffie-Hellman Key Agreement



Pairwise Diffie-Hellman Key Agreement

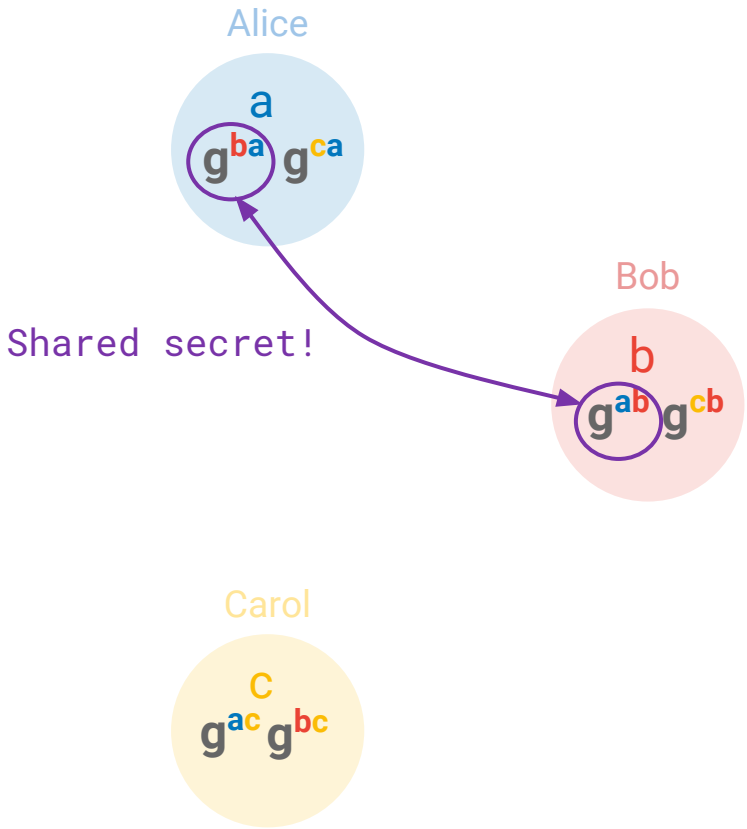


Pairwise Diffie-Hellman Key Agreement



Pairwise Diffie-Hellman Key Agreement

Secrets are scalars, but….

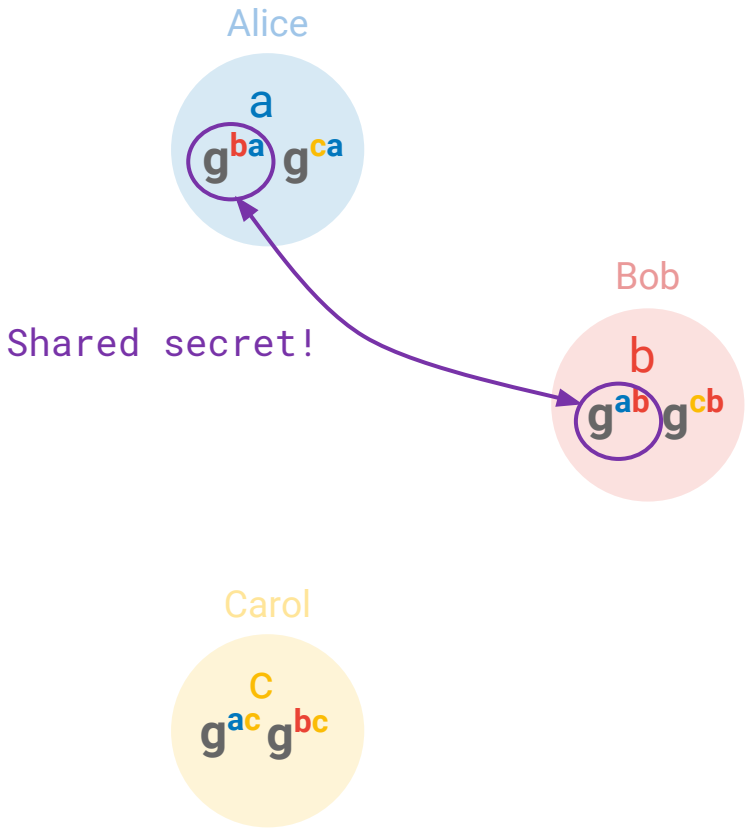


Pairwise Diffie-Hellman Key Agreement + PRNG Expansion

Secrets are scalars, but….

Use each secret to seed a **pseudorandom number generator**, generate paired antiparticle vectors.

PRNG(g^{ba}) → $\vec{\blacktriangledown} = -\vec{\blacktriangle}$

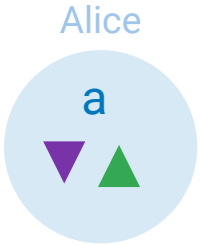


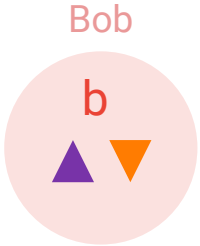
Pairwise Diffie-Hellman Key Agreement + PRNG Expansion

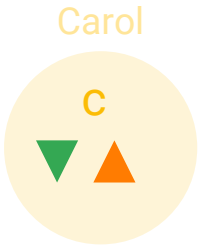
Secrets are scalars, but….

Use each secret to seed a pseudorandom number generator, generate paired antiparticle vectors.

PRNG(g^{ba}) → $\vec{\blacktriangledown} = -\vec{\blacktriangle}$





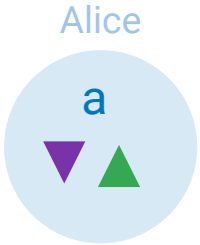


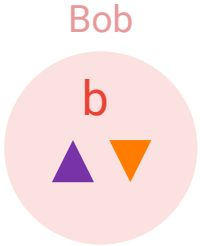
Pairwise Diffie-Hellman Key Agreement + PRNG Expansion

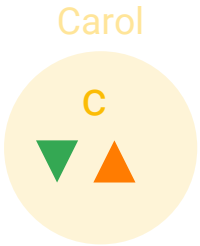
Secrets are scalars, but....

Use each secret to seed a pseudorandom number generator, generate paired antiparticle vectors.

PRNG(g^{ba}) → $\vec{\blacktriangledown} = -\vec{\blacktriangle}$







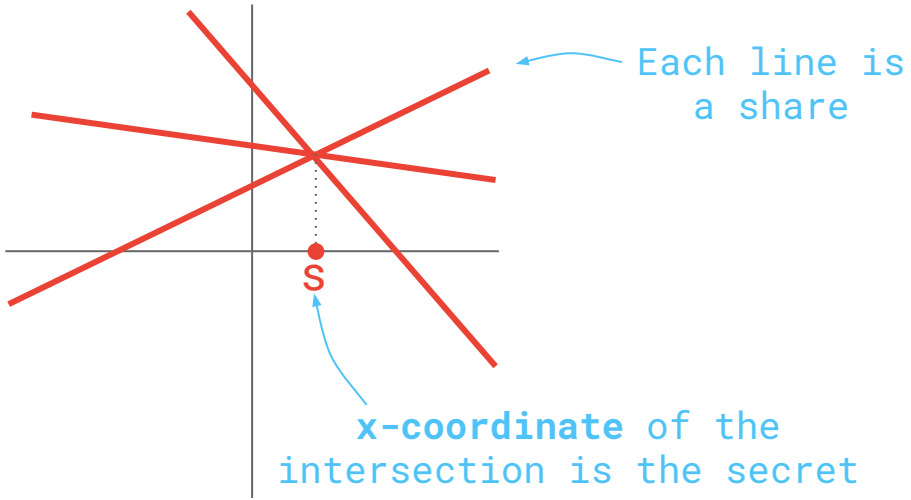
1. **Efficiency via pseudorandom generator**
2. **Mobile phones typically don't support peer-to-peer communication anyhow.**
3. **Fewer secrets = easier recovery.**

k-out-of-*n* Threshold Secret Sharing

Goal: Break a secret into *n* pieces, called shares.

- **<k shares:** learn nothing
- **≥k shares**: recover *s* perfectly.

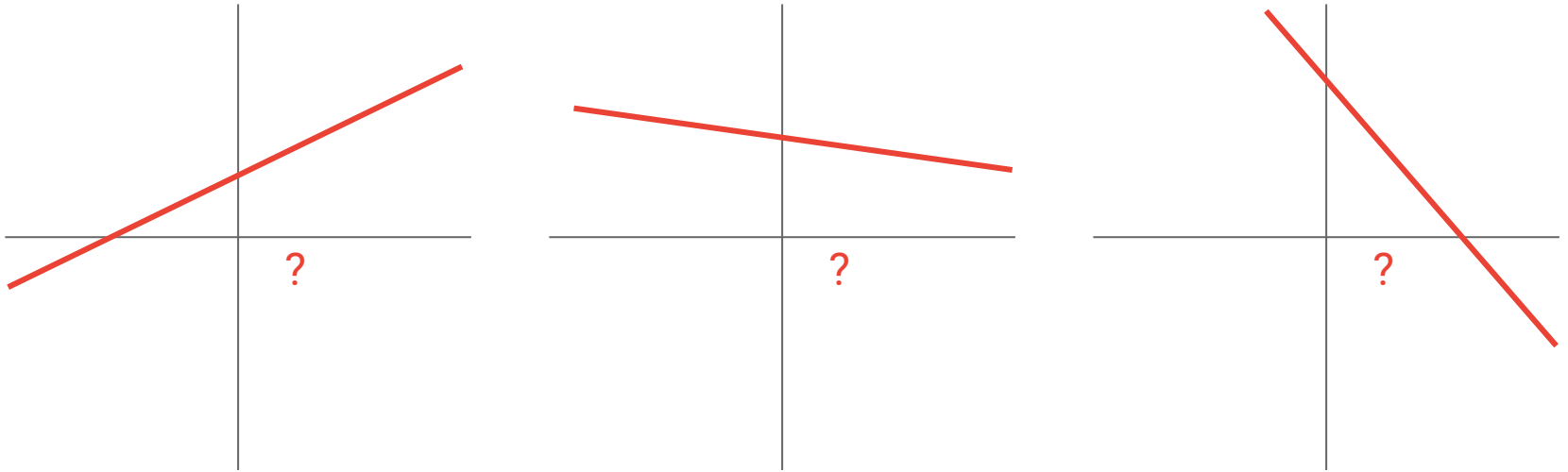
2-out-of-3 secret sharing:



k-out-of-*n* Threshold Secret Sharing

Goal: Break a secret into *n* pieces, called shares.

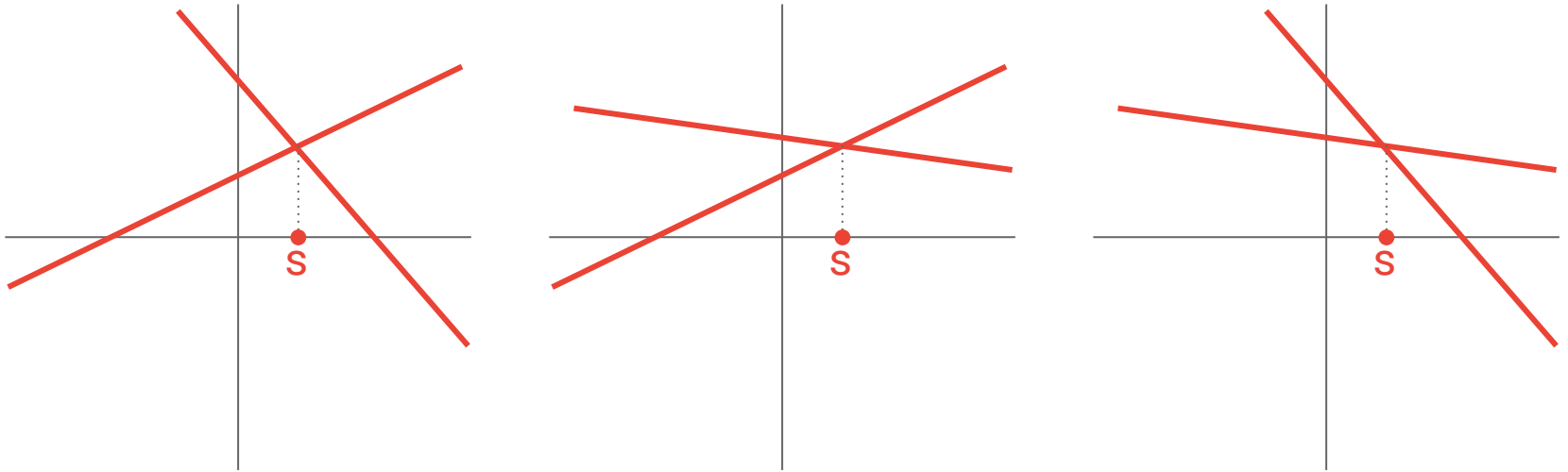
- **<k shares:** learn nothing
- **≥k shares**: recover *s* perfectly



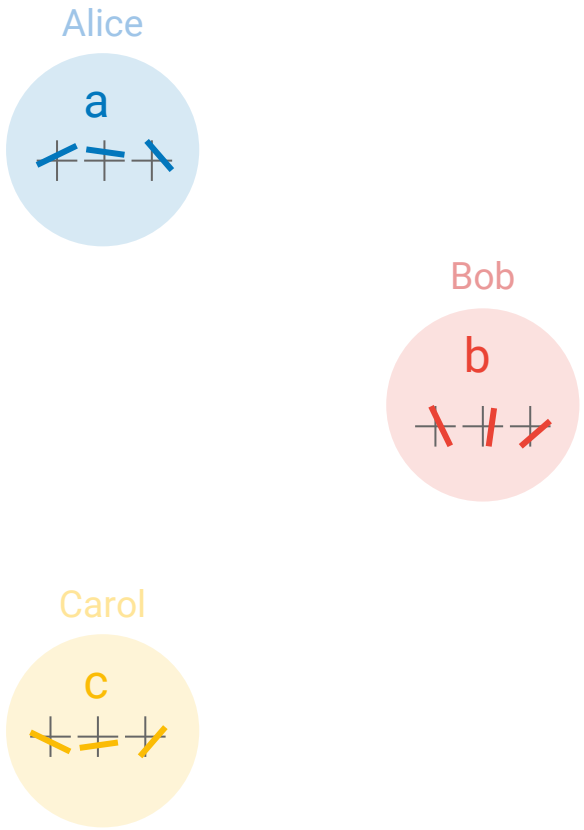
k-out-of-*n* Threshold Secret Sharing

Goal: Break a secret into *n* pieces, called shares.

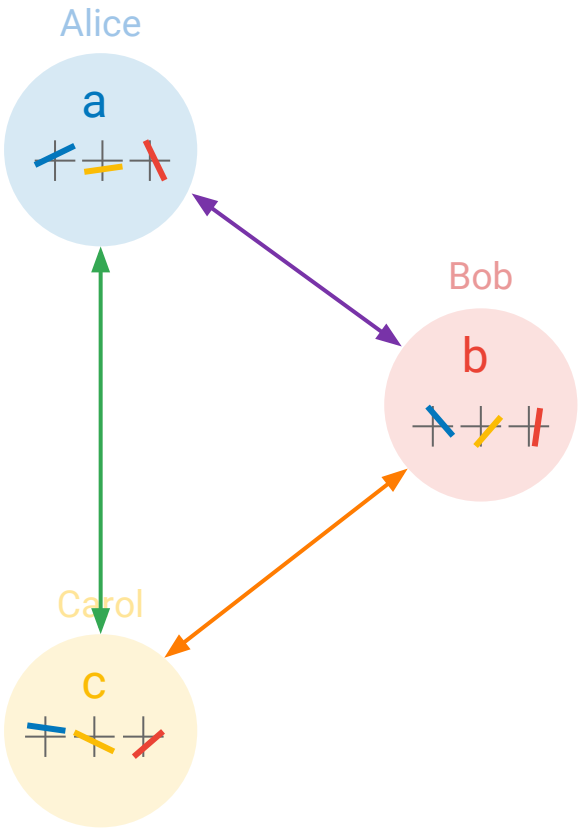
- **<k shares:** learn nothing
- **≥k shares**: recover *s* perfectly

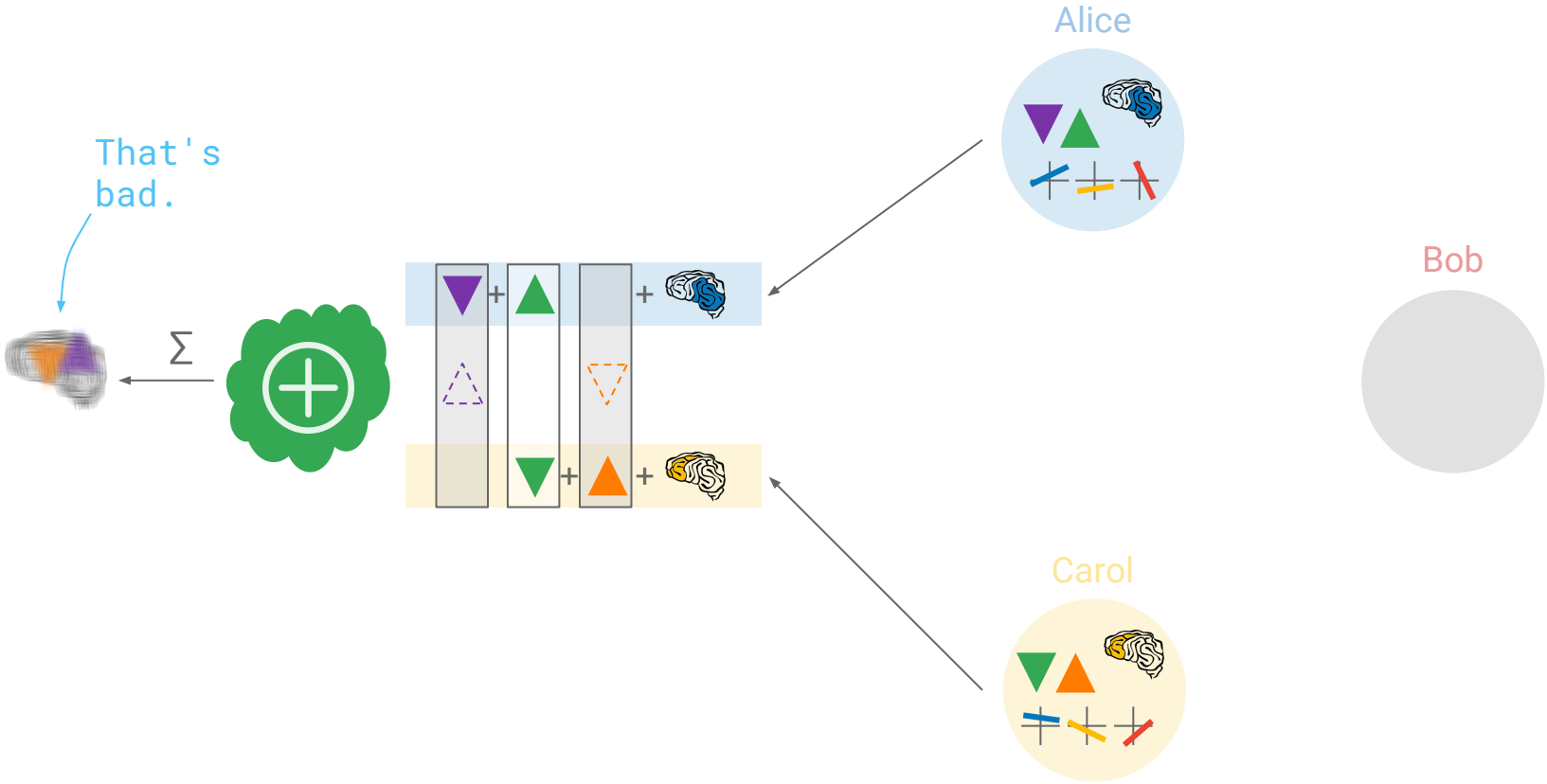


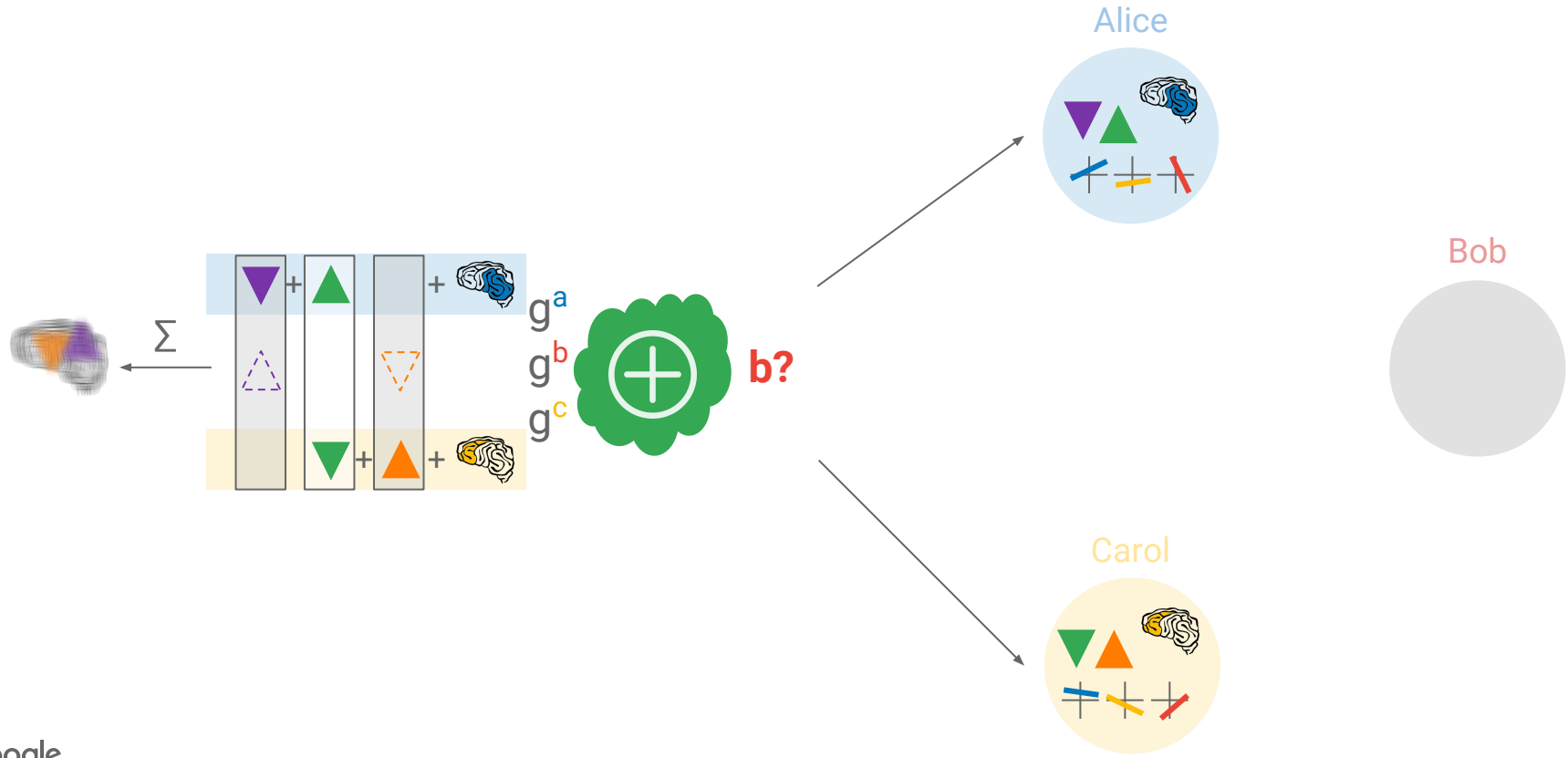
Users make shares of their secrets

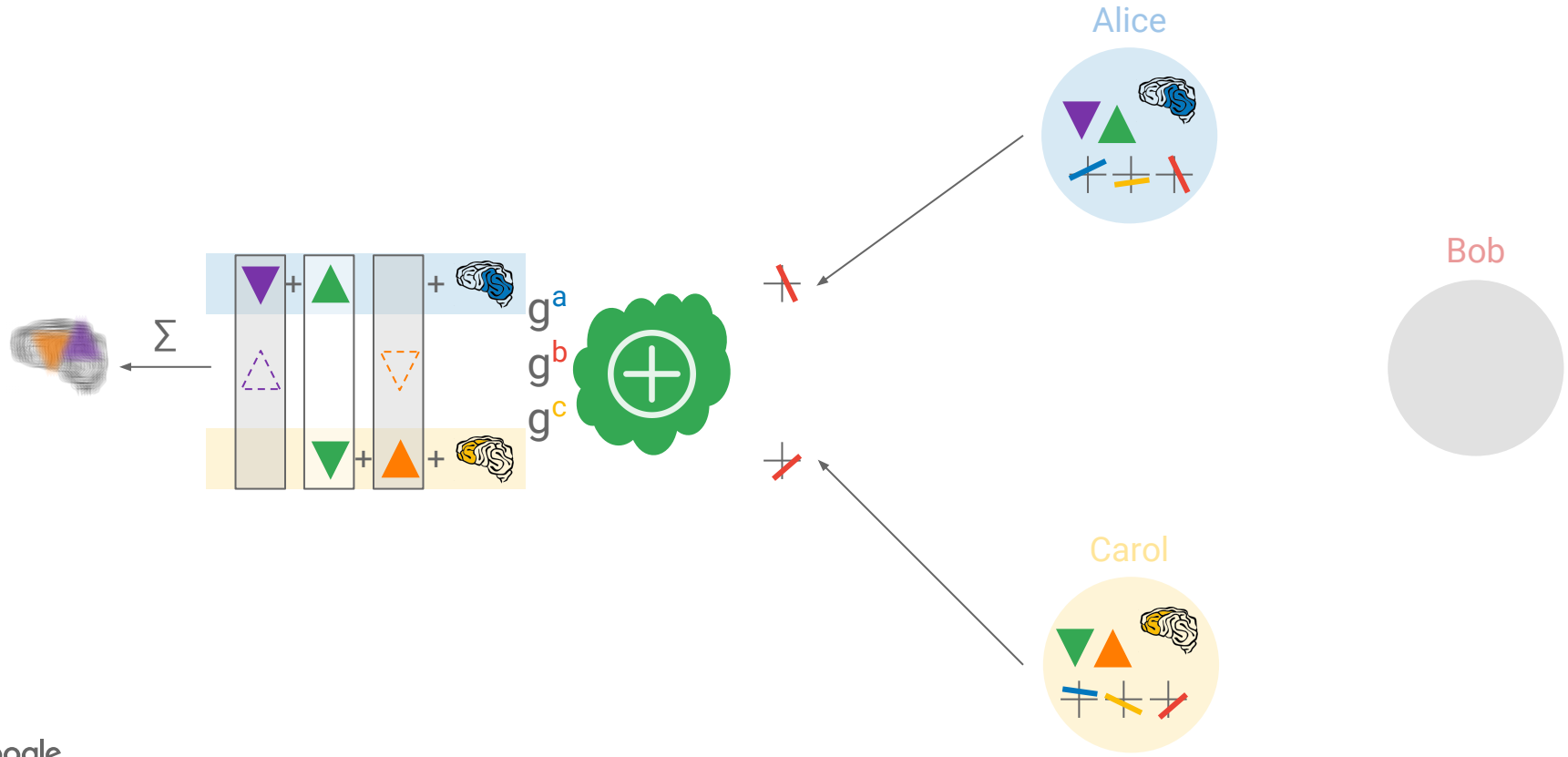


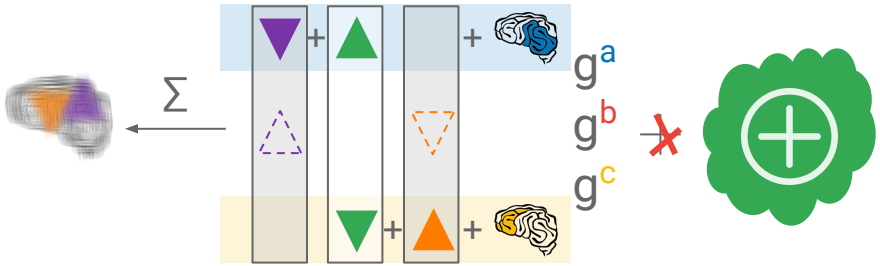
And exchange with their peers



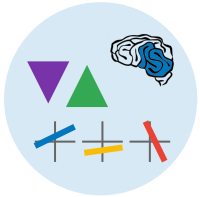

That's bad.
Σ
Alice
Bob
Carol





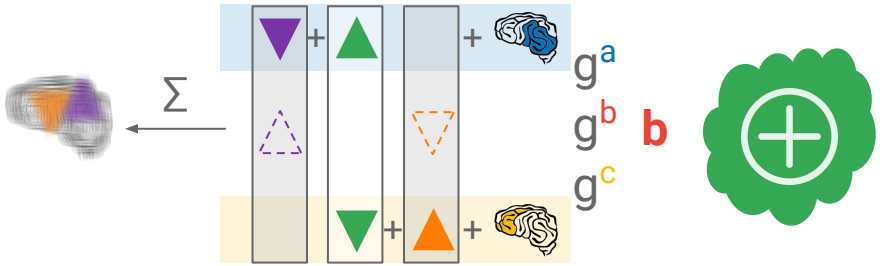


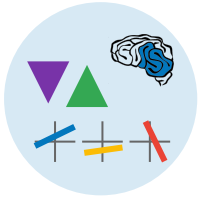
Alice

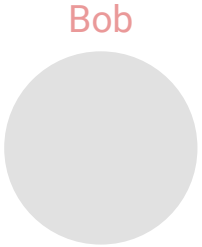


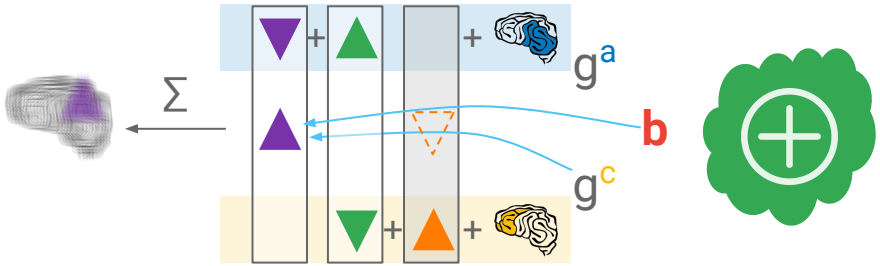
Bob

Carol

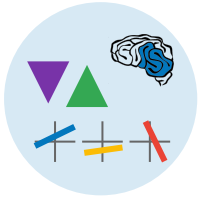








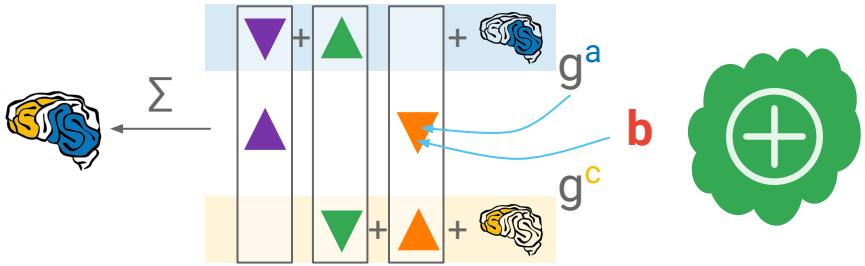
Alice



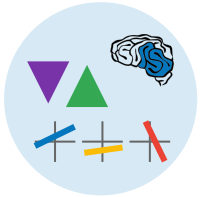
Bob

Carol

Google



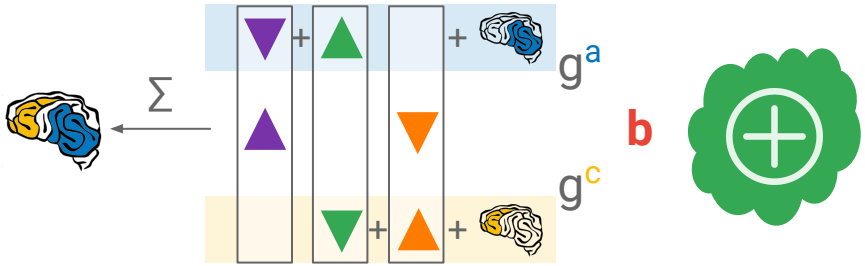
Alice



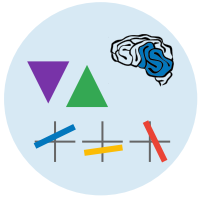
Bob

Carol

Google



Alice

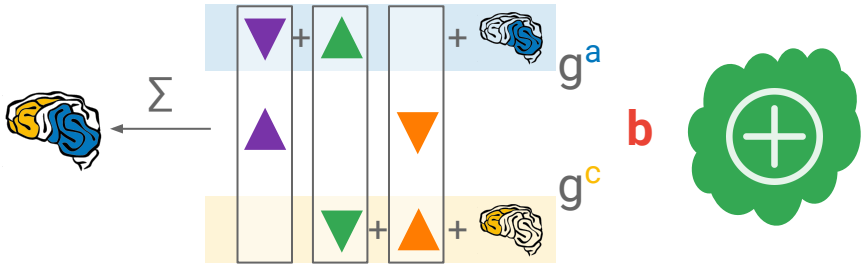


Bob

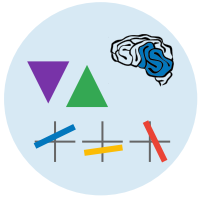
Carol

Enough honest users + a high enough threshold
⇒ dishonest users cannot reconstruct the secret.

Google



Alice

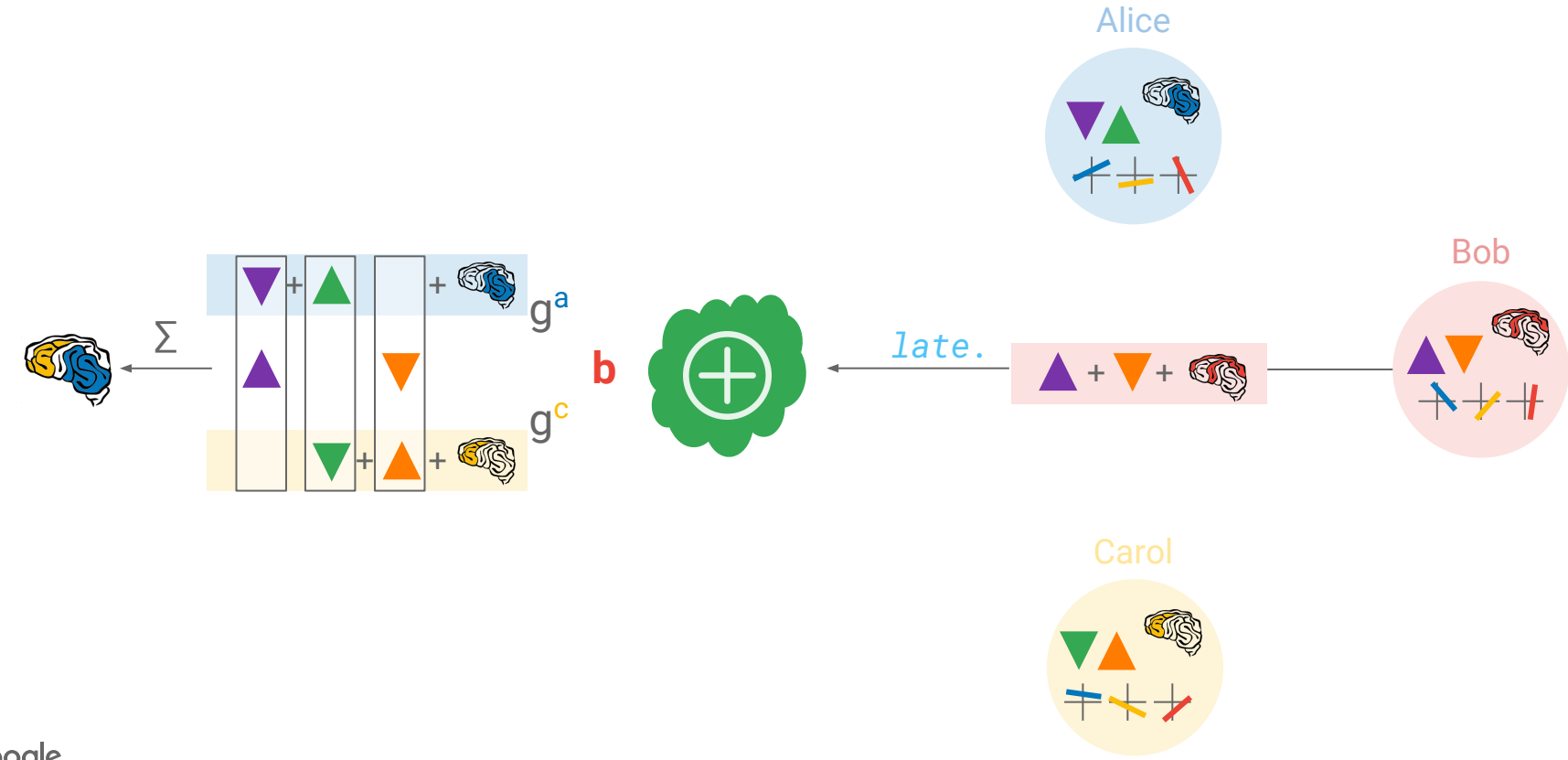


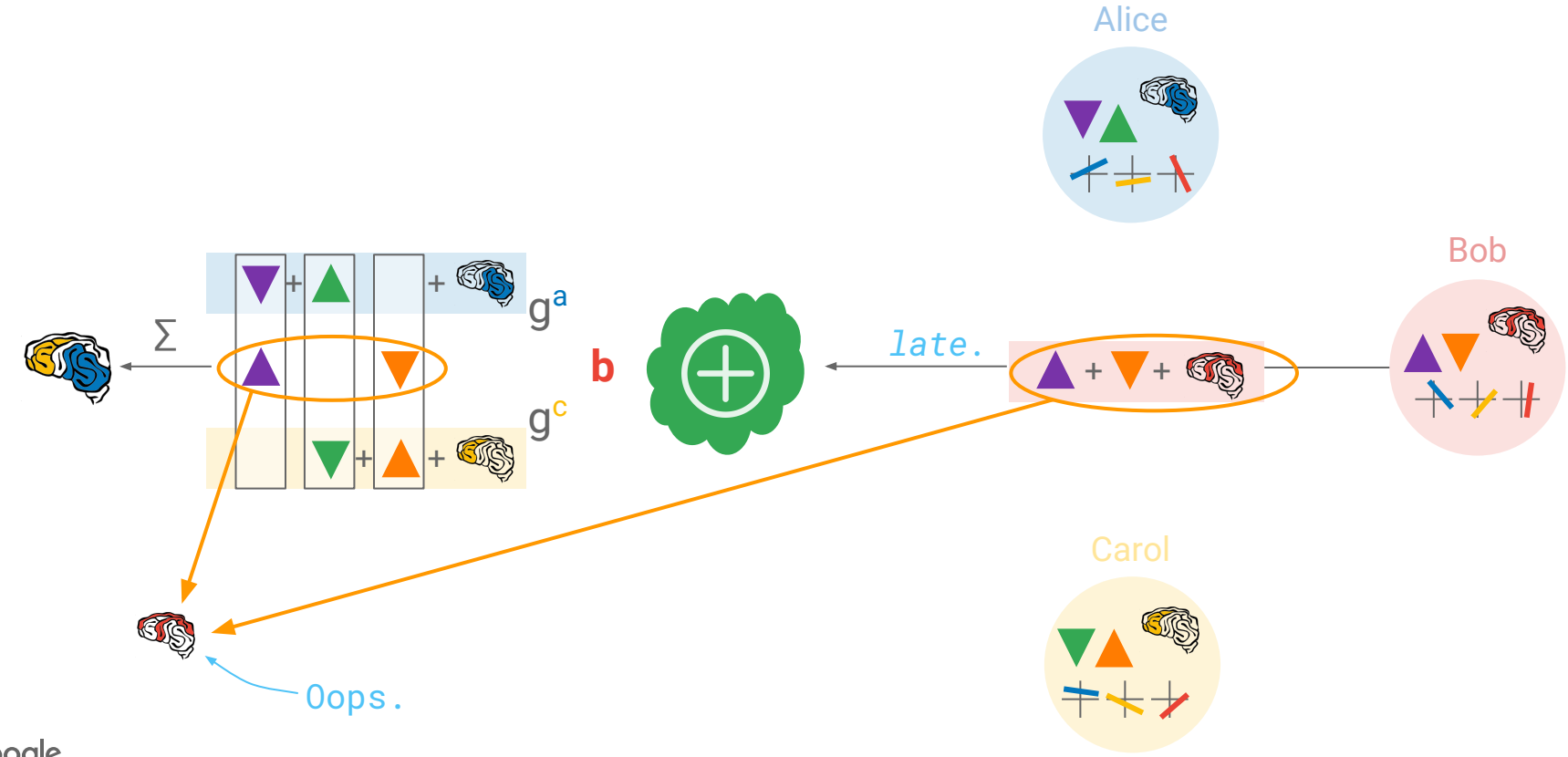
Bob

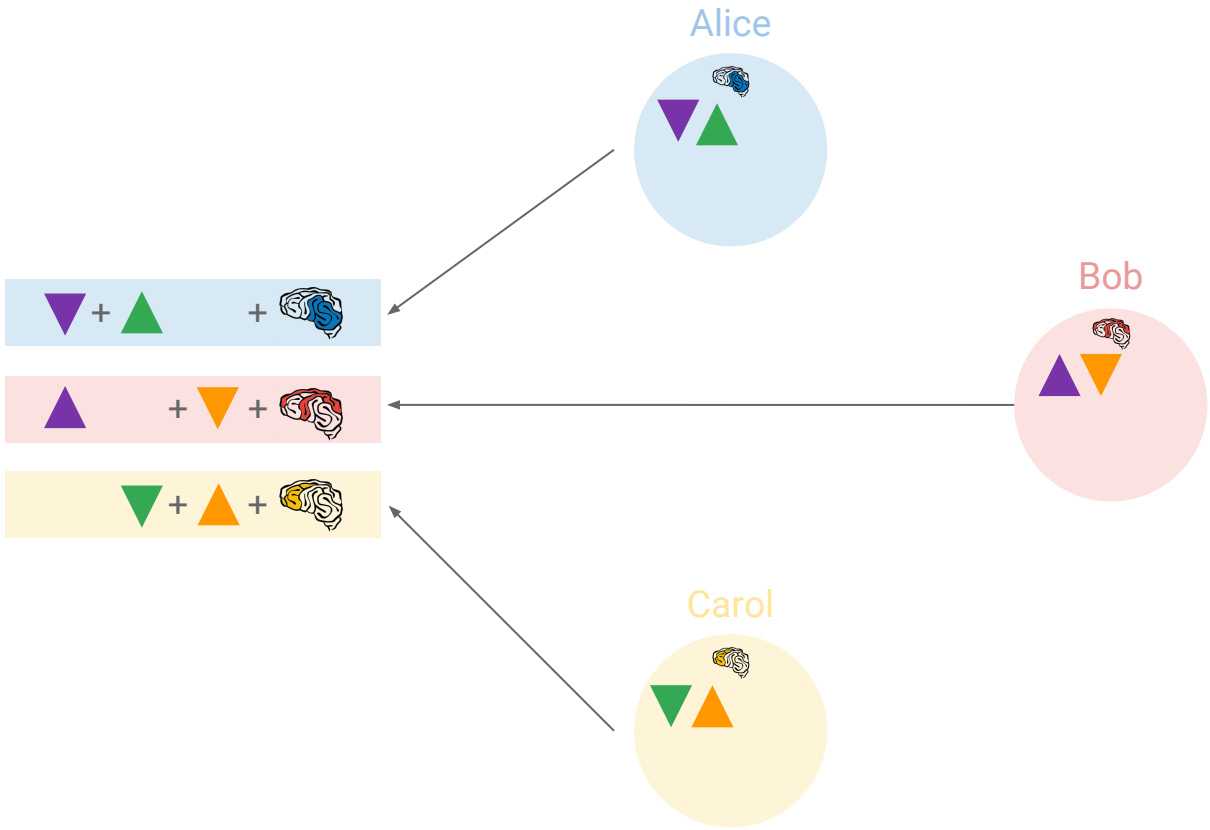
Carol

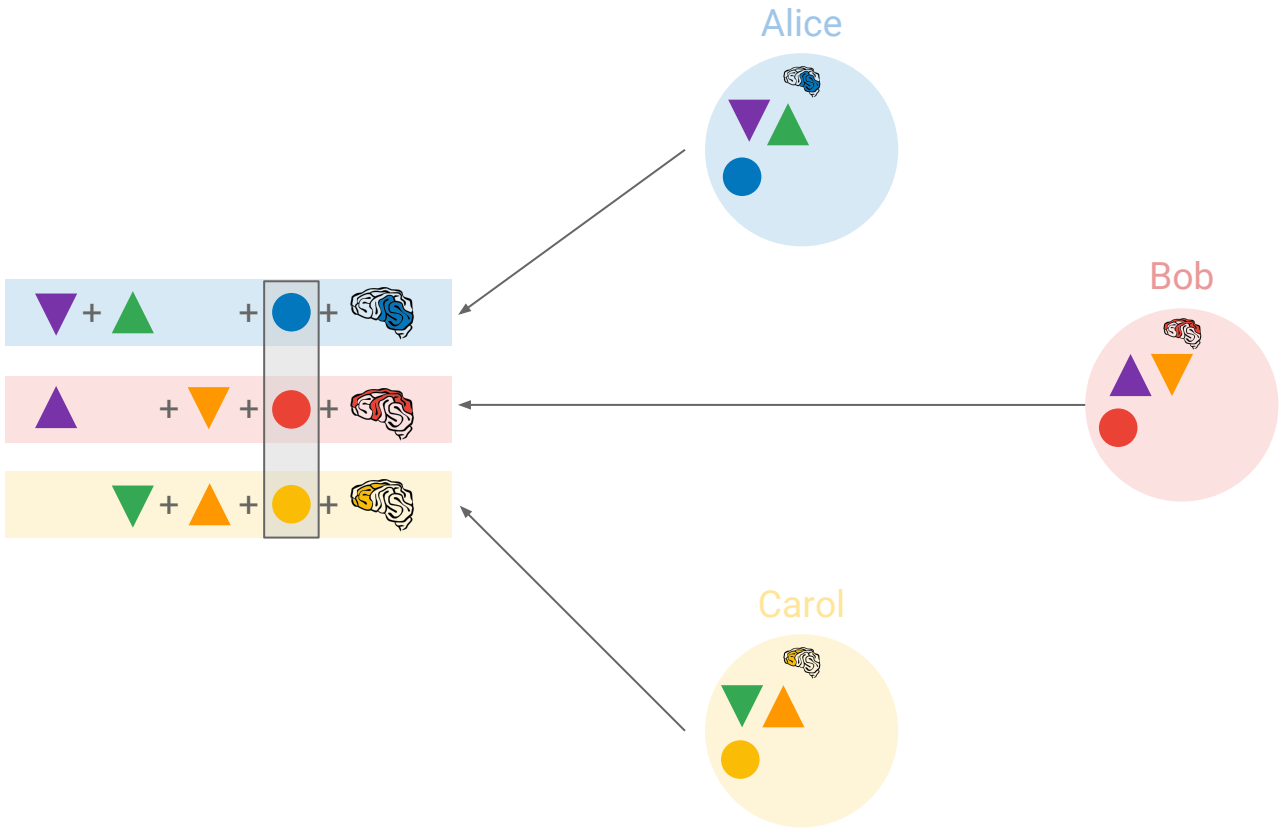
Enough honest users + a high enough threshold
⇒ dishonest users cannot reconstruct the secret.

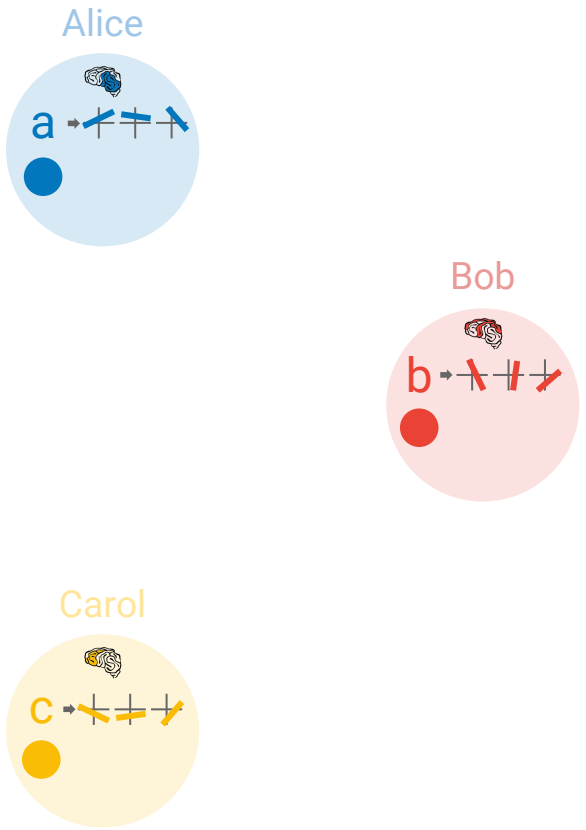
However....



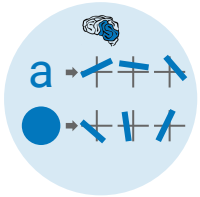

Alice
Bob
Carol
late.
b
Σ
g^a
g^c
Oops.


Alice
Bob
Carol

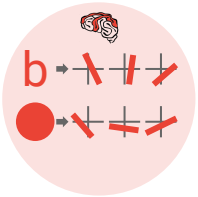

Alice
Bob
Carol



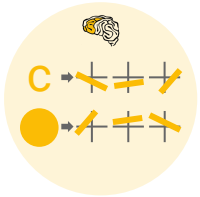
Alice

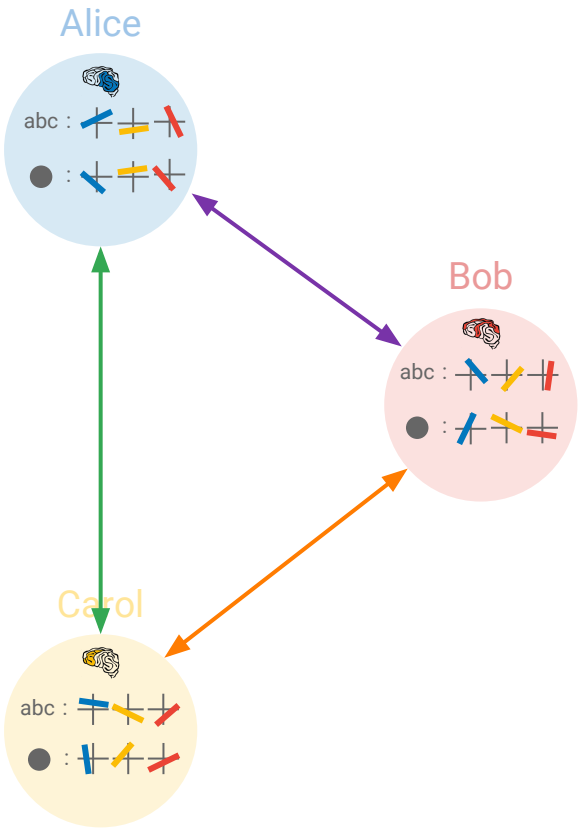


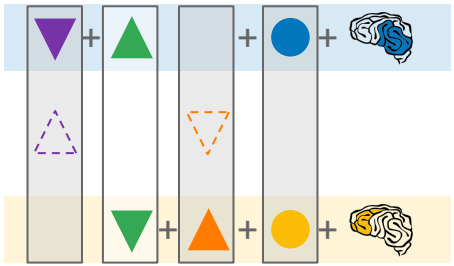
Bob



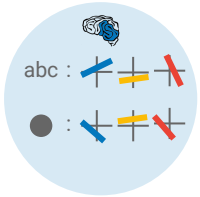
Carol




Alice
abc :
:
Bob
abc :
:
Carol
abc :
:

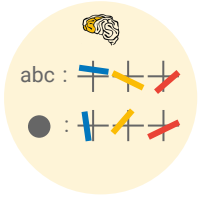


Alice

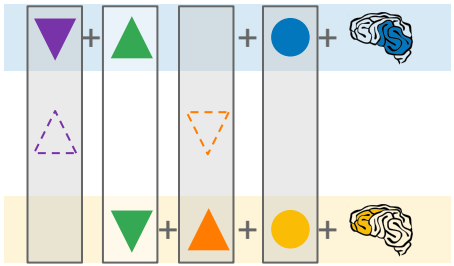


Bob

Carol

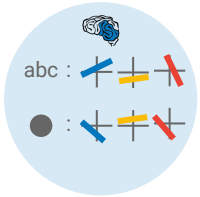


Google



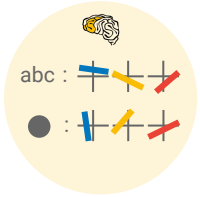
(●, b, ●)?

Alice

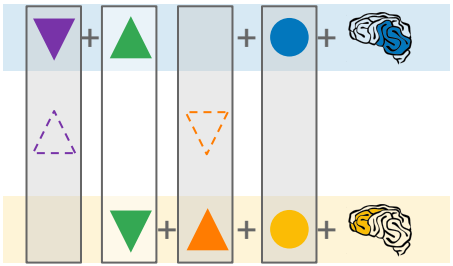


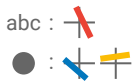
Bob

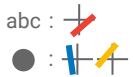
Carol

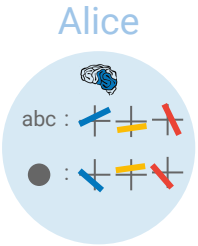


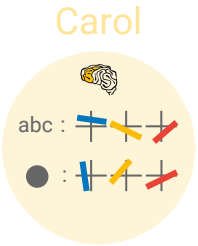
Google

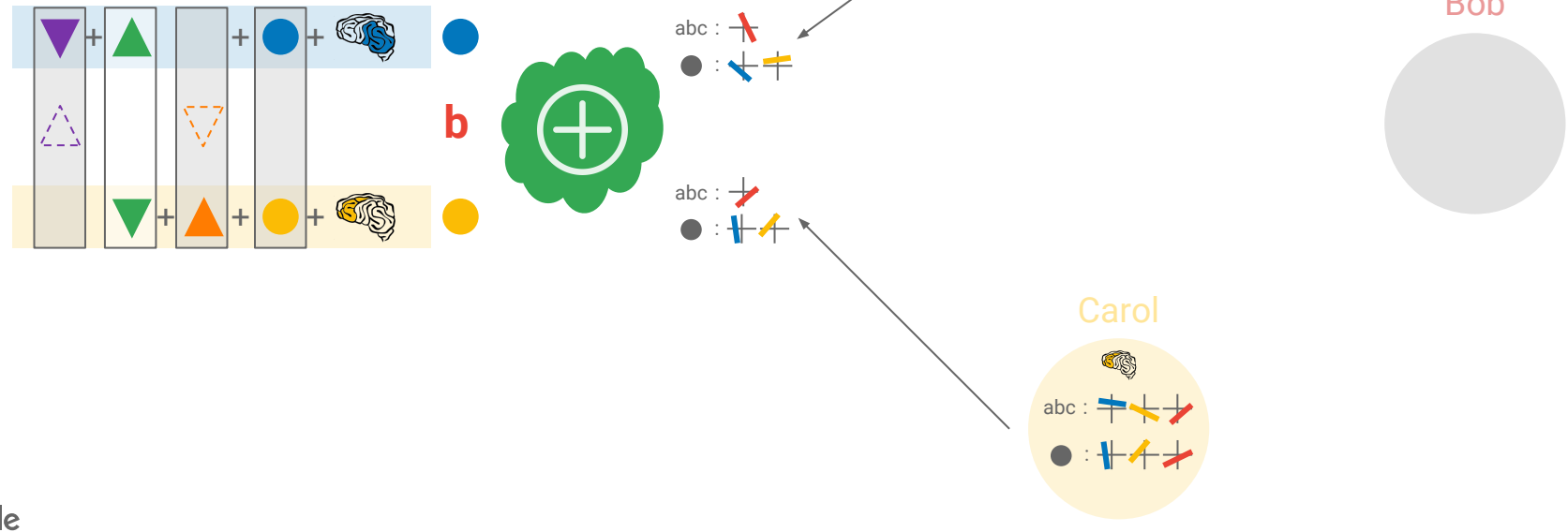


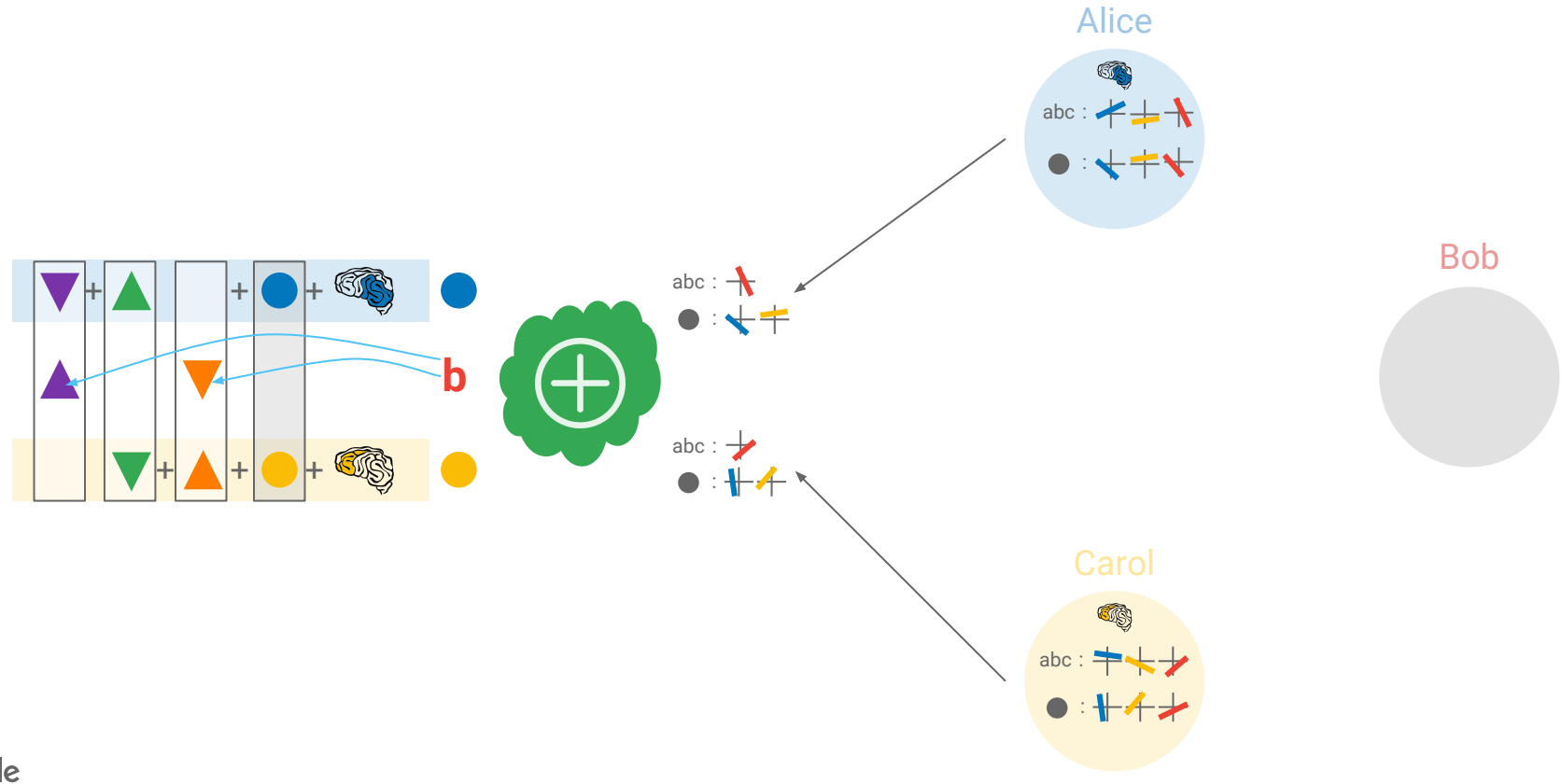




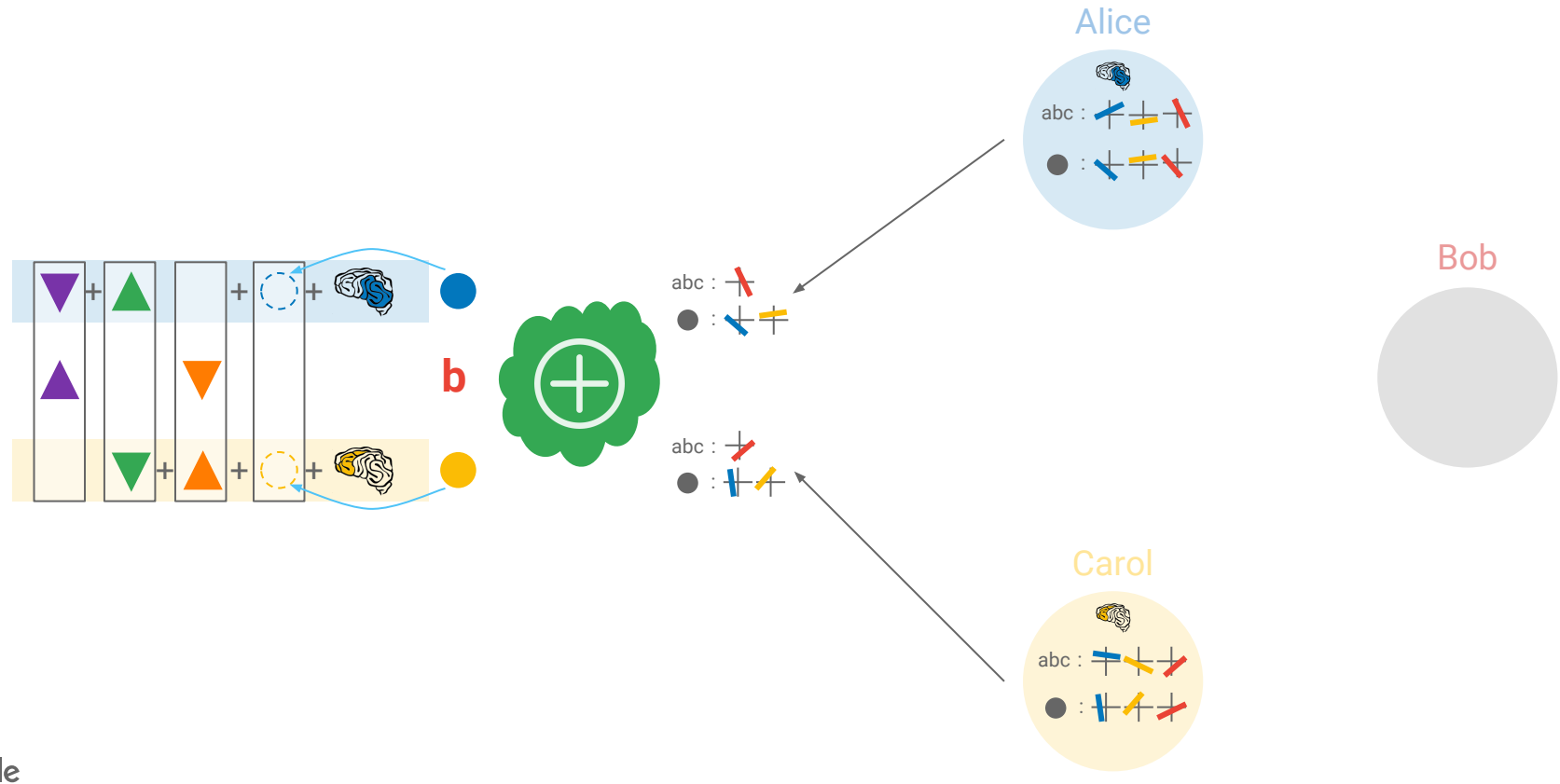


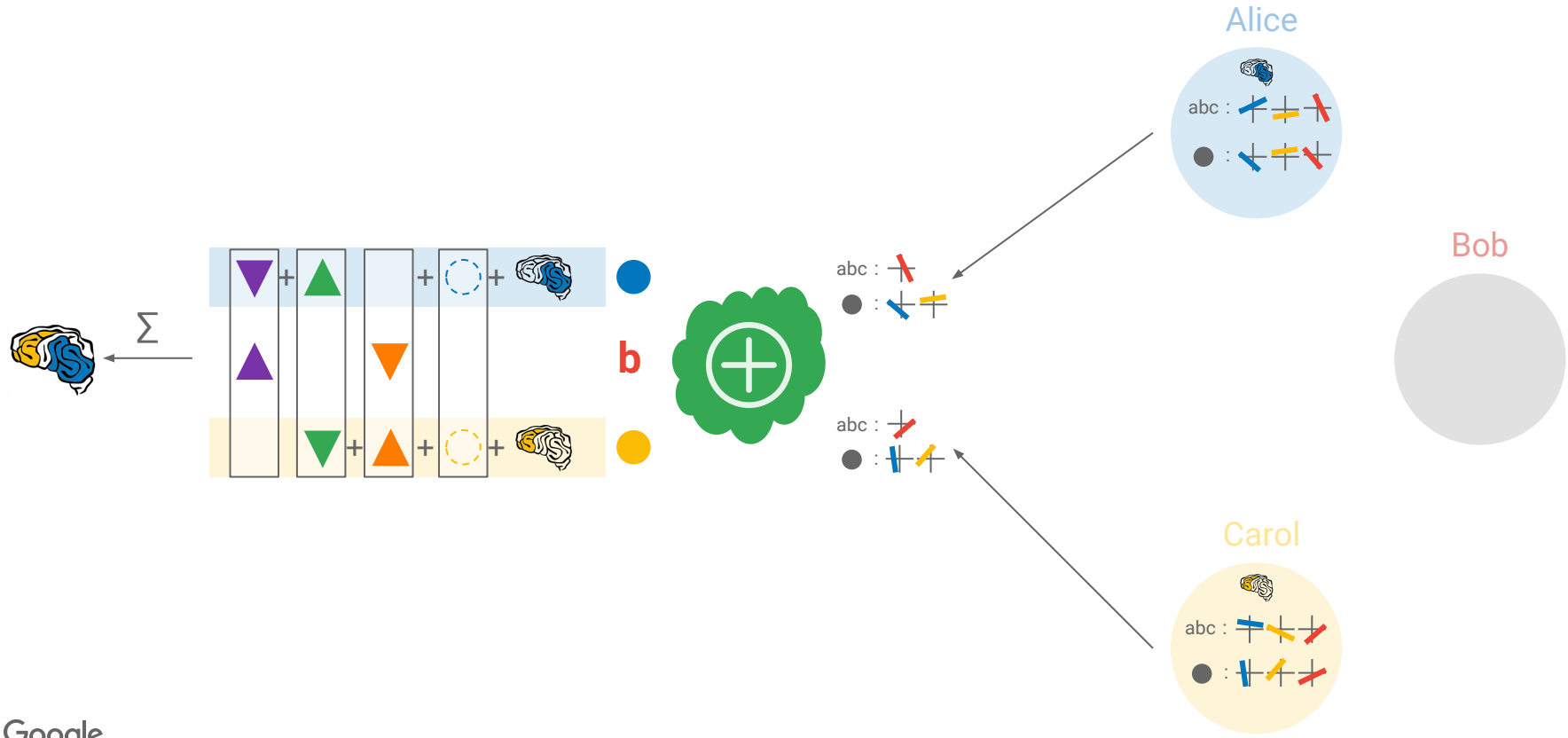



Alice
abc :
:
Bob
abc :
:
b
abc :
:
Carol
abc :
:

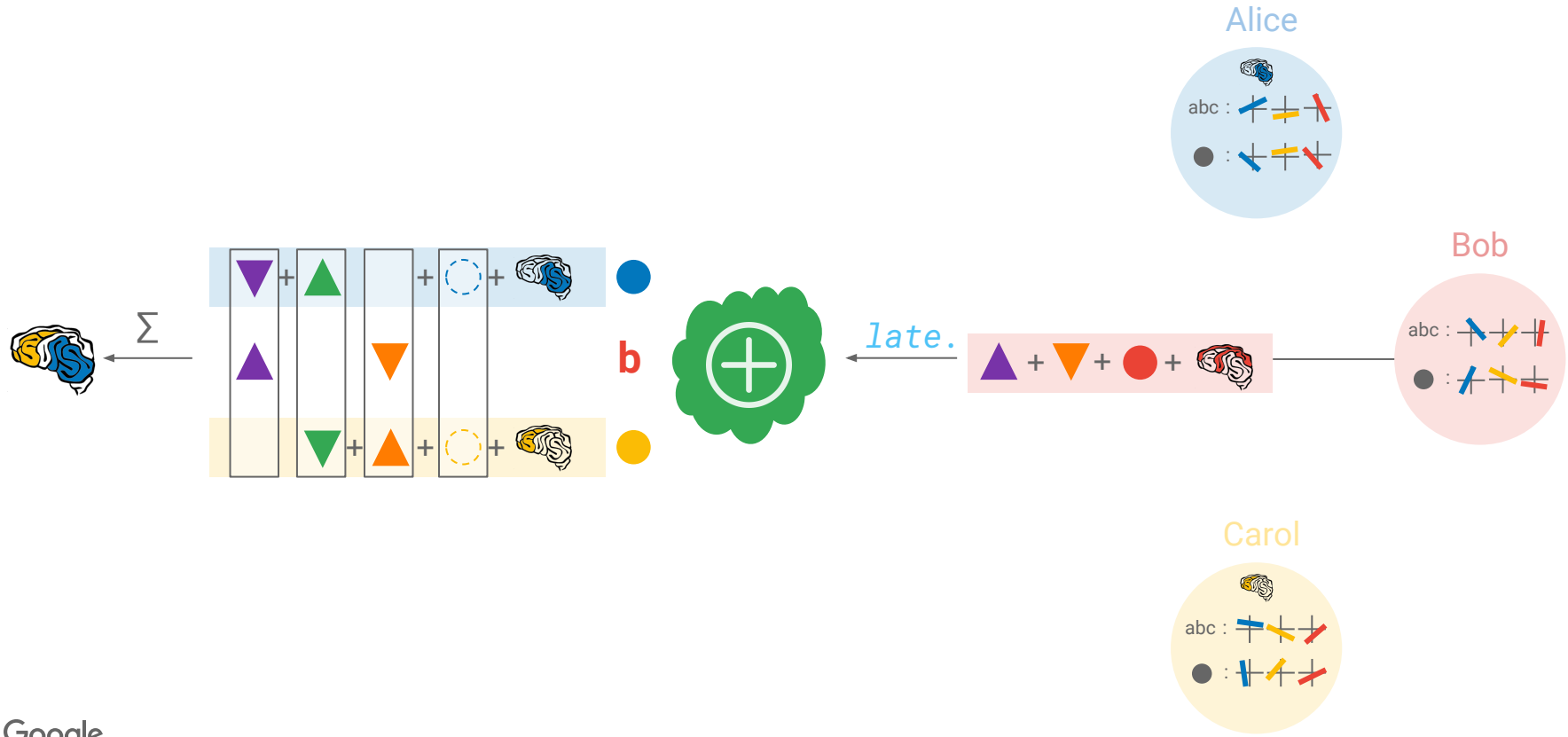


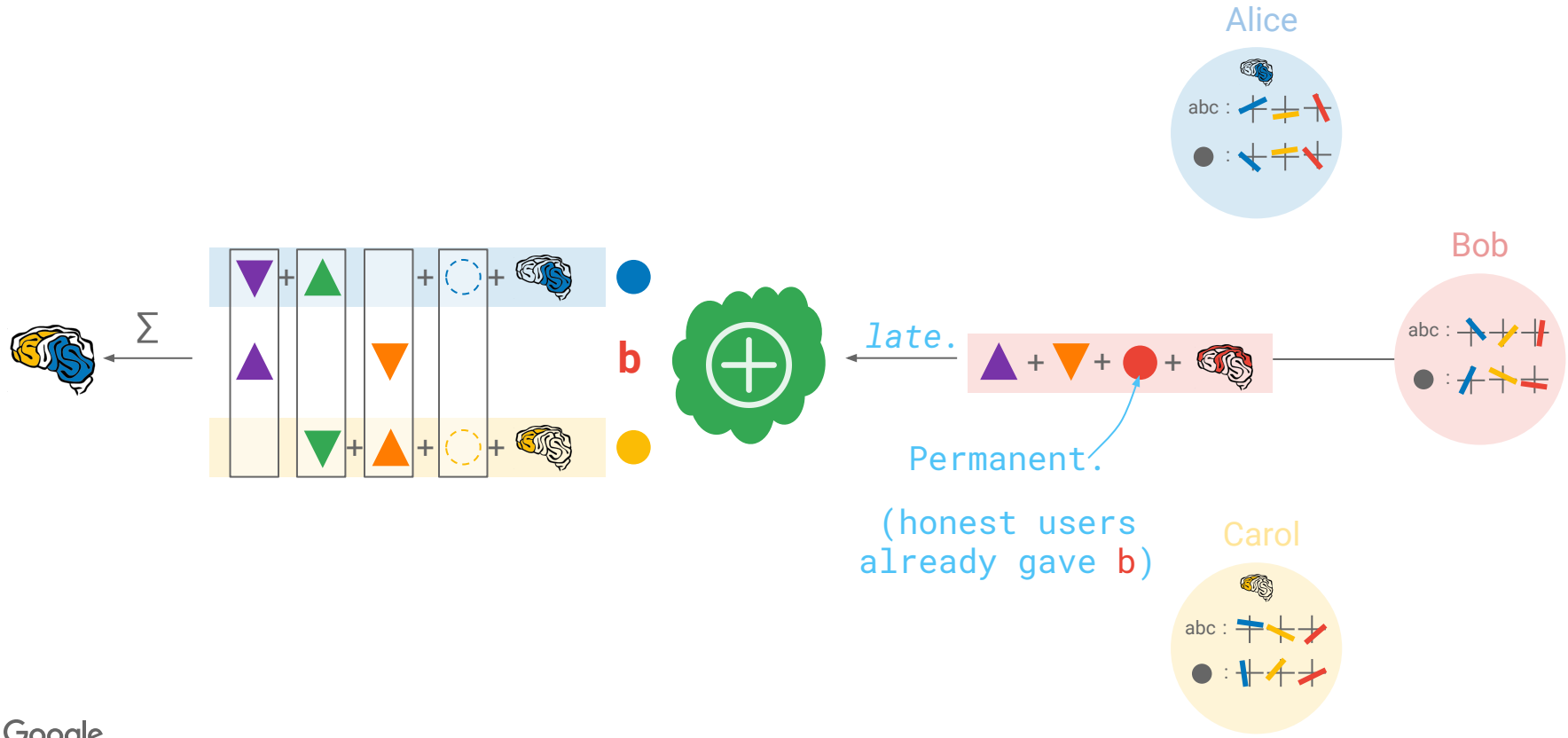
Google


Alice
abc :
:
Bob
abc :
:
b
abc :
:
Carol
abc :
:



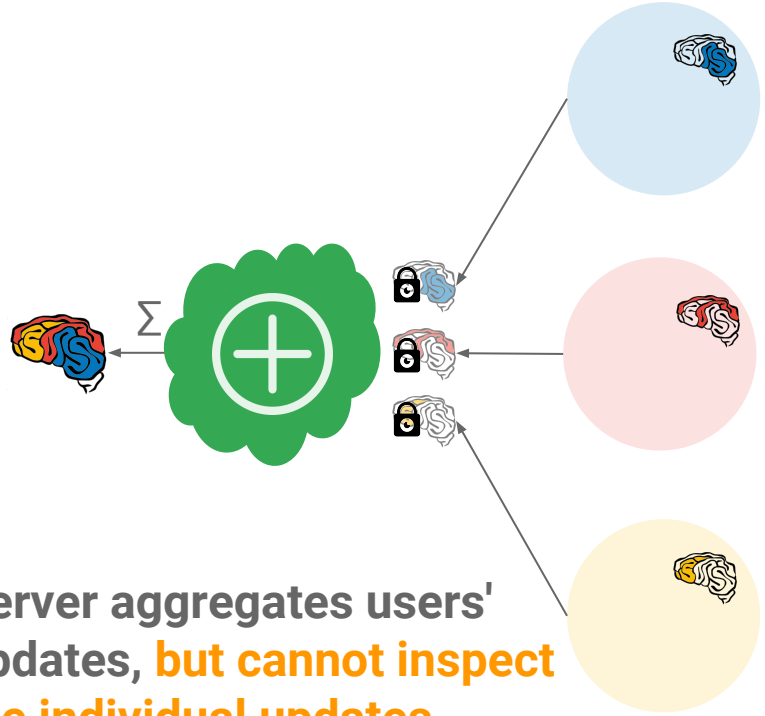
Google


Alice
abc :
:
Bob
abc :
:
Carol
abc :
:
late.
b
Σ


Alice
abc :
:
Bob
abc :
:
Carol
abc :
:
Σ
b
late.
Permanent.
(honest users
already gave b)

Secure Aggregation

K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, K. Seth.
Practical Secure Aggregation for Privacy-Preserving Machine Learning. *To appear at CCS 2017.*



Server aggregates users' updates, but cannot inspect the individual updates.

Google

Interactive Cryptographic Protocol

Each phase, 1000 clients + server interchange messages over 4 rounds of communication.

Secure	Robust
⅓ malicious clients + fully observed server	⅓ clients can drop out

Communication Efficient

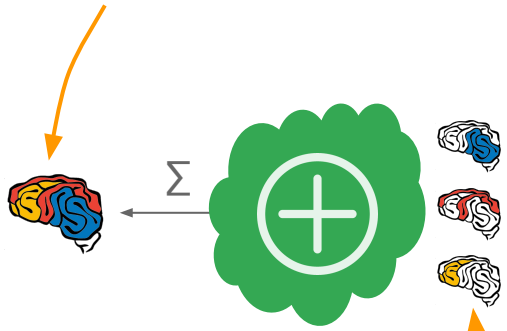
# Params	Bits/Param	# Users	Expansion
2^{20} = 1 m	16	2^{10} = 1 k	**1.73x**
2^{24} = 16 m	16	2^{14} = 16 k	**1.98x**

This Talk

1. Why Federated Learning?
2. FL vs. other distributed optimization
3. FL Algorithms
4. Minimizing Communication
5. Secure Aggregation
6. **Differential Privacy**

Federated Learning

Might the final model memorize a user's data?



Might these updates contain privacy-sensitive data?

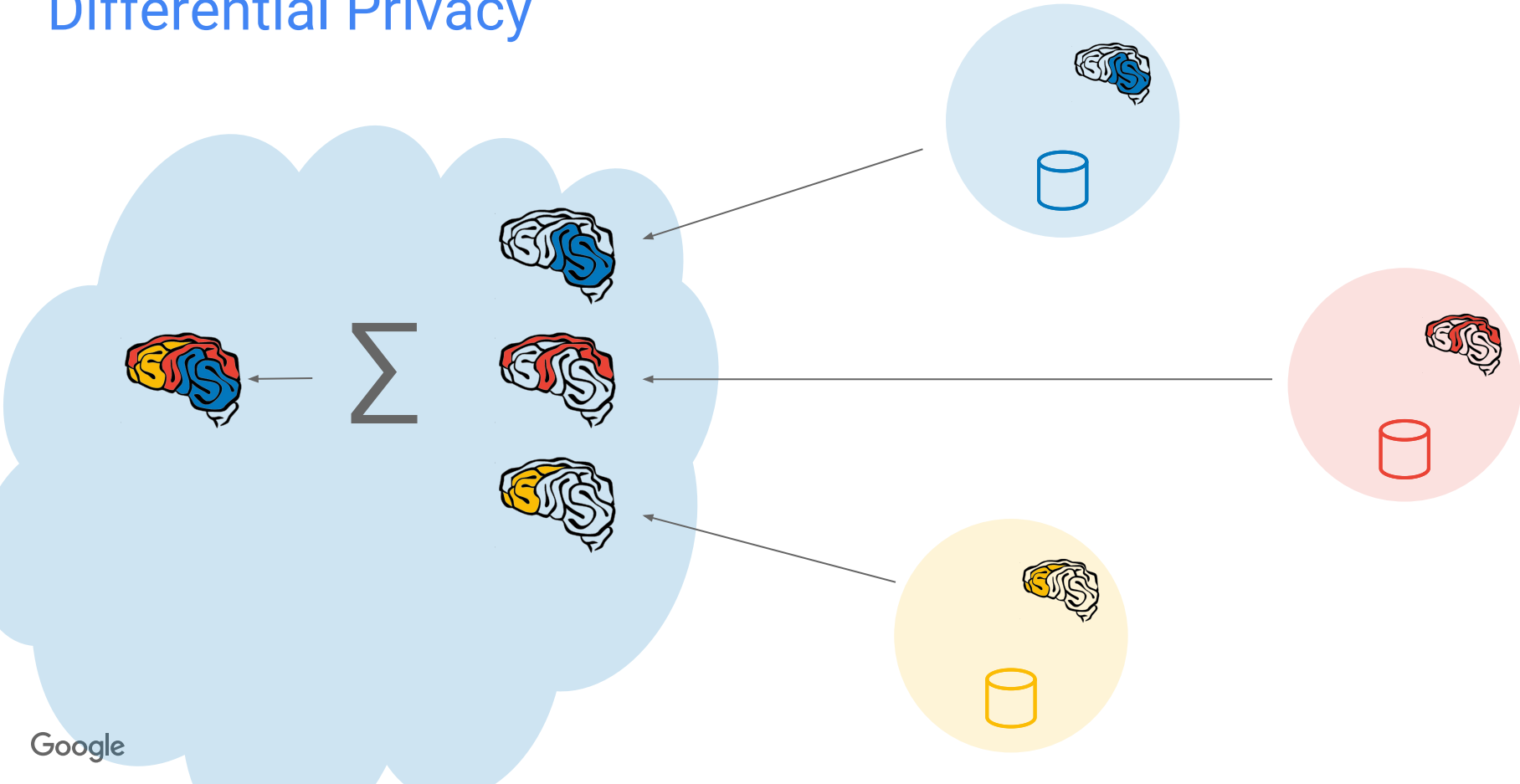
1. Ephemeral
2. Focused
3. Only in Aggregate
4. **Differential Privacy**

Differential Privacy

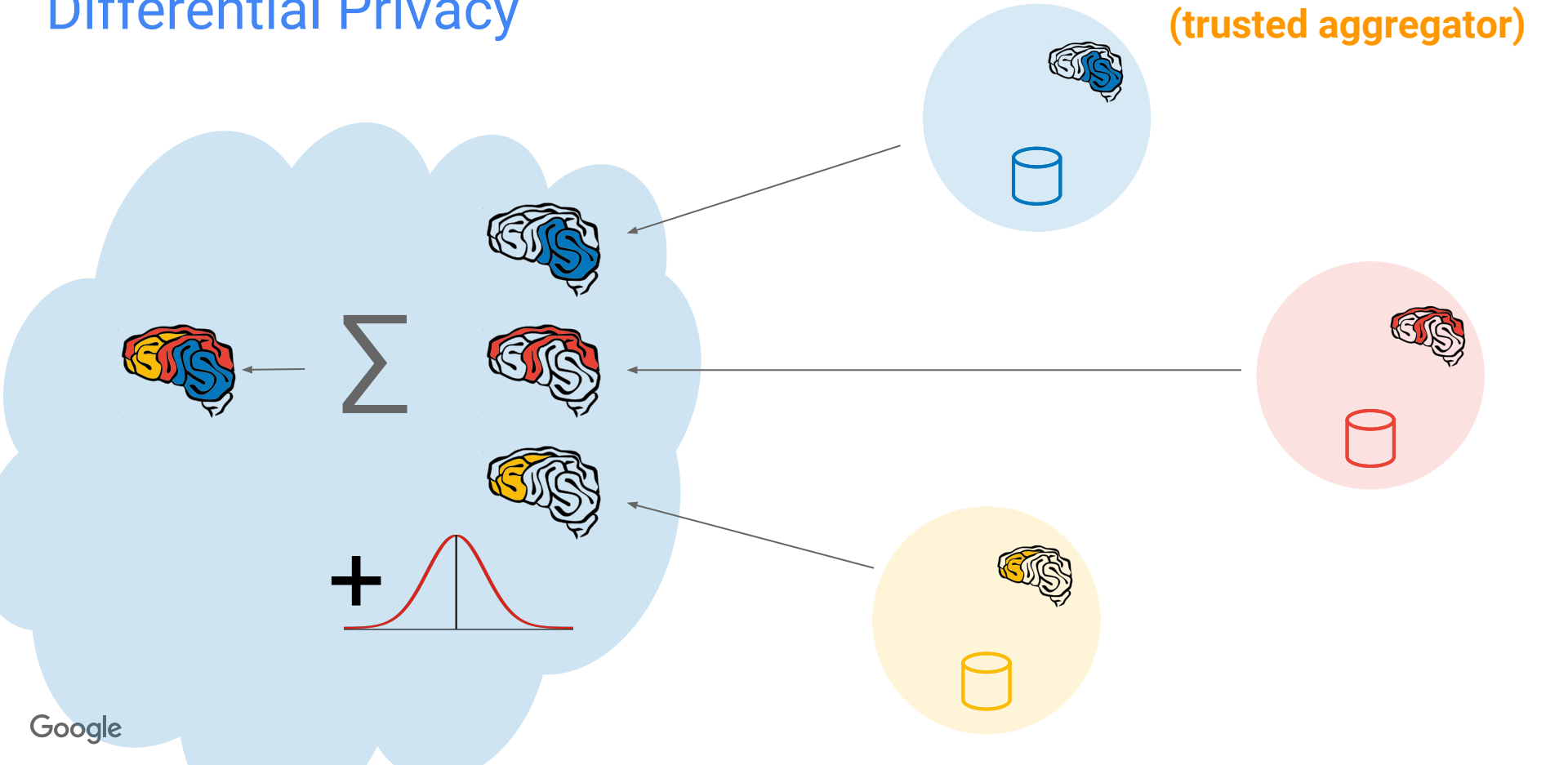
Differential privacy is the statistical science of trying to learn **as much as possible about a group** while learning **as little as possible about any individual in it.**

Andy Greenberg
Wired 2016.06.13

Differential Privacy



Differential Privacy



Federated Averaging

Server

Until Converged:

1. Select a random subset (e.g. C=100) of the (online) clients

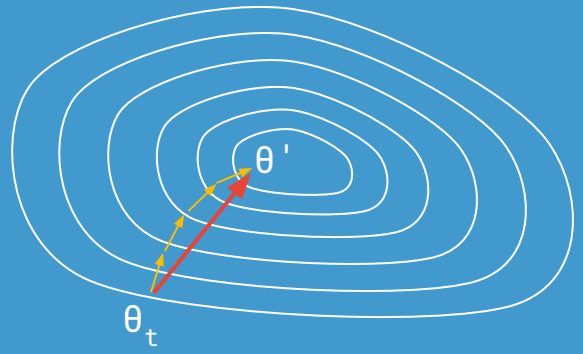
2. In parallel, send current parameters θ_t to those clients

Selected Client k

1. Receive θ_t from server.

2. Run some number of minibatch SGD steps, producing θ'

3. Return $\theta' - \theta_t$ to server.



3. $\theta_{t+1} = \theta_t +$ data-weighted average of client updates

Differentially-Private Federated Averaging

McMahan, Ramage, Talwar, Zhang. **Learning Differentially Private Recurrent Language Models**.

Server

Until Converged:

1. Select each user **independently** with **probability q**, for say E[C]=1000 clients

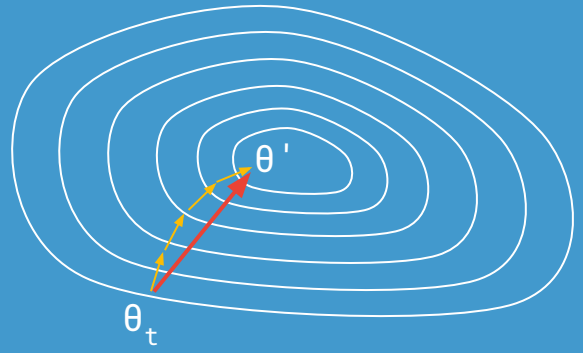
2. In parallel, send current parameters θ_t to those clients

Selected Client k

1. Receive θ_t from server.

2. Run some number of minibatch SGD steps, producing θ'

3. Return $\theta' - \theta_t$ to server.



3. $\theta_{t+1} = \theta_t +$ data-weighted average of client updates

Differentially-Private Federated Averaging

McMahan, Ramage, Talwar, Zhang. **Learning Differentially Private Recurrent Language Models**.

Server

Until Converged:

1. Select each user **independently** with **probability q**, for say E[C]=1000 clients

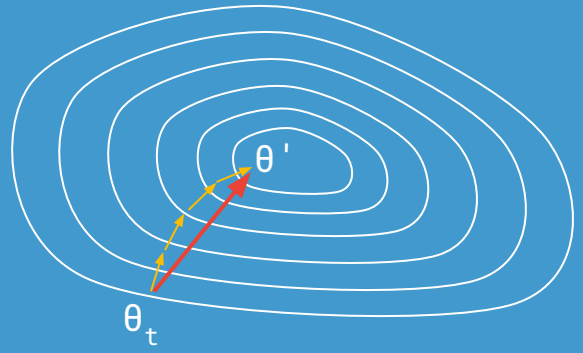
2. In parallel, send current parameters θ_t to those clients

Selected Client k

1. Receive θ_t from server.

2. Run some number of minibatch SGD steps, producing θ'

3. Return **Clip(θ'-θ_t)** to server.



3. θ_{t+1} = θ_t + data-weighted average of client updates

Differentially-Private Federated Averaging

McMahan, Ramage, Talwar, Zhang. **Learning Differentially Private Recurrent Language Models**.

Server

Until Converged:

1. Select each user **independently** with **probability q**, for say E[C]=1000 clients

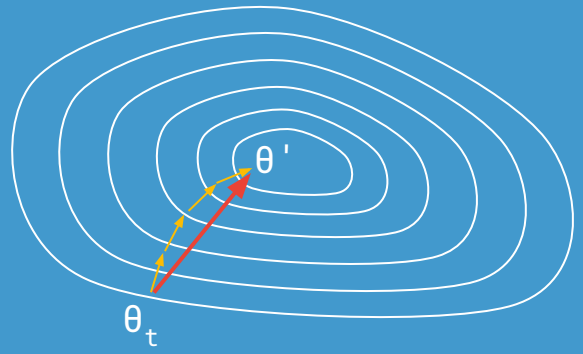
2. In parallel, send current parameters θ_t to those clients

Selected Client k

1. Receive θ_t from server.

2. Run some number of minibatch SGD steps, producing θ'

3. Return **Clip($\theta'-\theta_t$)** to server.



3. $\theta_{t+1} = \theta_t +$ **bounded sensitivity** data-weighted average of client updates

Differentially-Private Federated Averaging

McMahan, Ramage, Talwar, Zhang. **Learning Differentially Private Recurrent Language Models**.

Server

Until Converged:

1. Select each user **independently** with **probability q**, for say E[C]=1000 clients

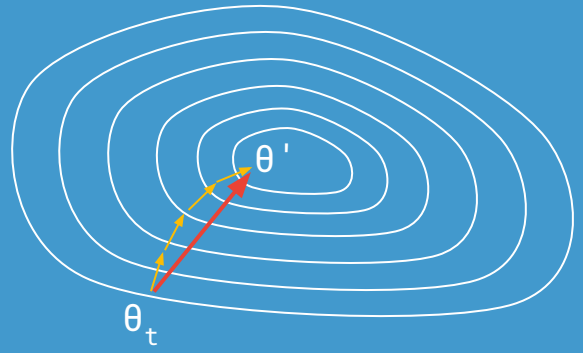
2. In parallel, send current parameters θ_t to those clients

Selected Client k

1. Receive θ_t from server.

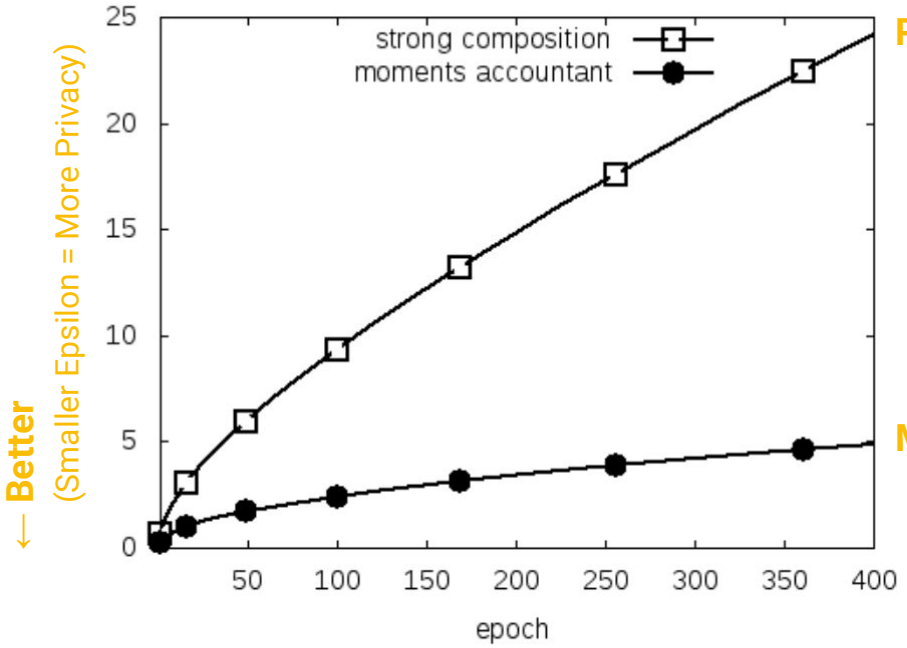
2. Run some number of minibatch SGD steps, producing θ'

3. Return **Clip($\theta'-\theta_t$)** to server.



3. $\theta_{t+1} = \theta_t +$ **bounded sensitivity** data-weighted average of client updates + Gaussian noise **N(0, Iσ²)**

Privacy Accounting for Noisy SGD: Moments Accountant

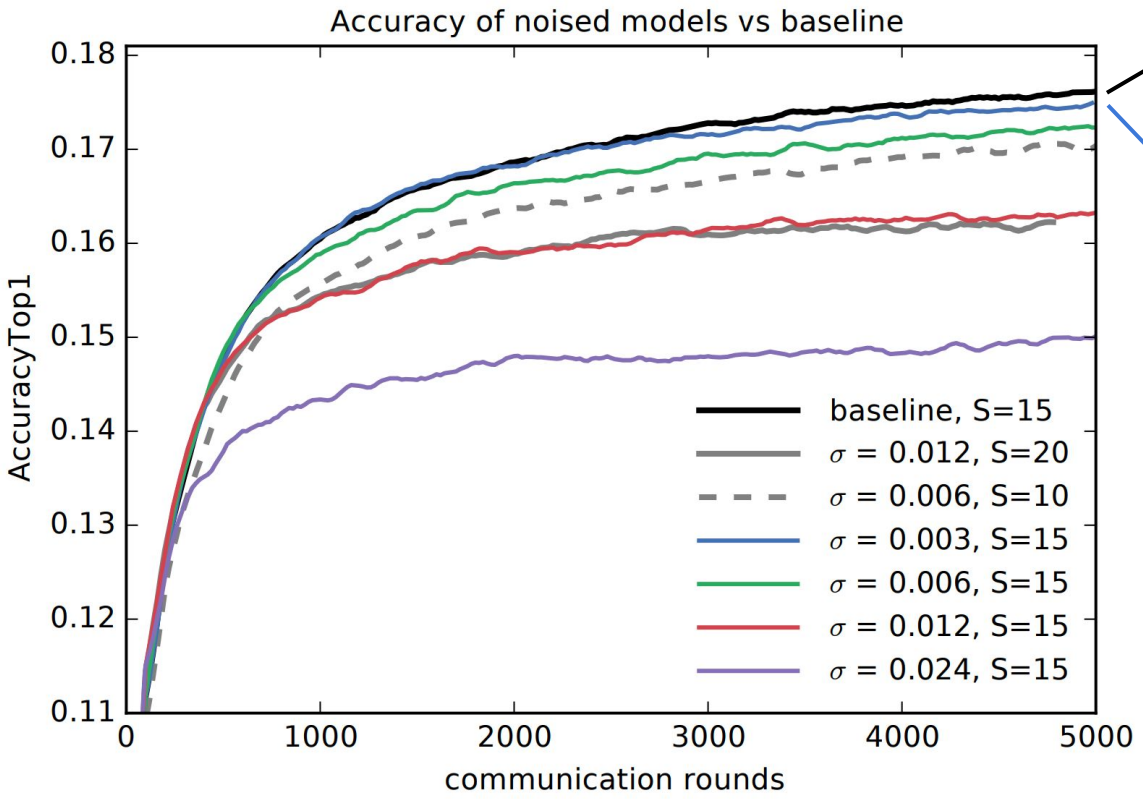


Previous composition theorems

Moments Accountant

M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, & L. Zhang. **Deep Learning with Differential Privacy**. CCS 2016.

User Differential Privacy for Federated Language Models



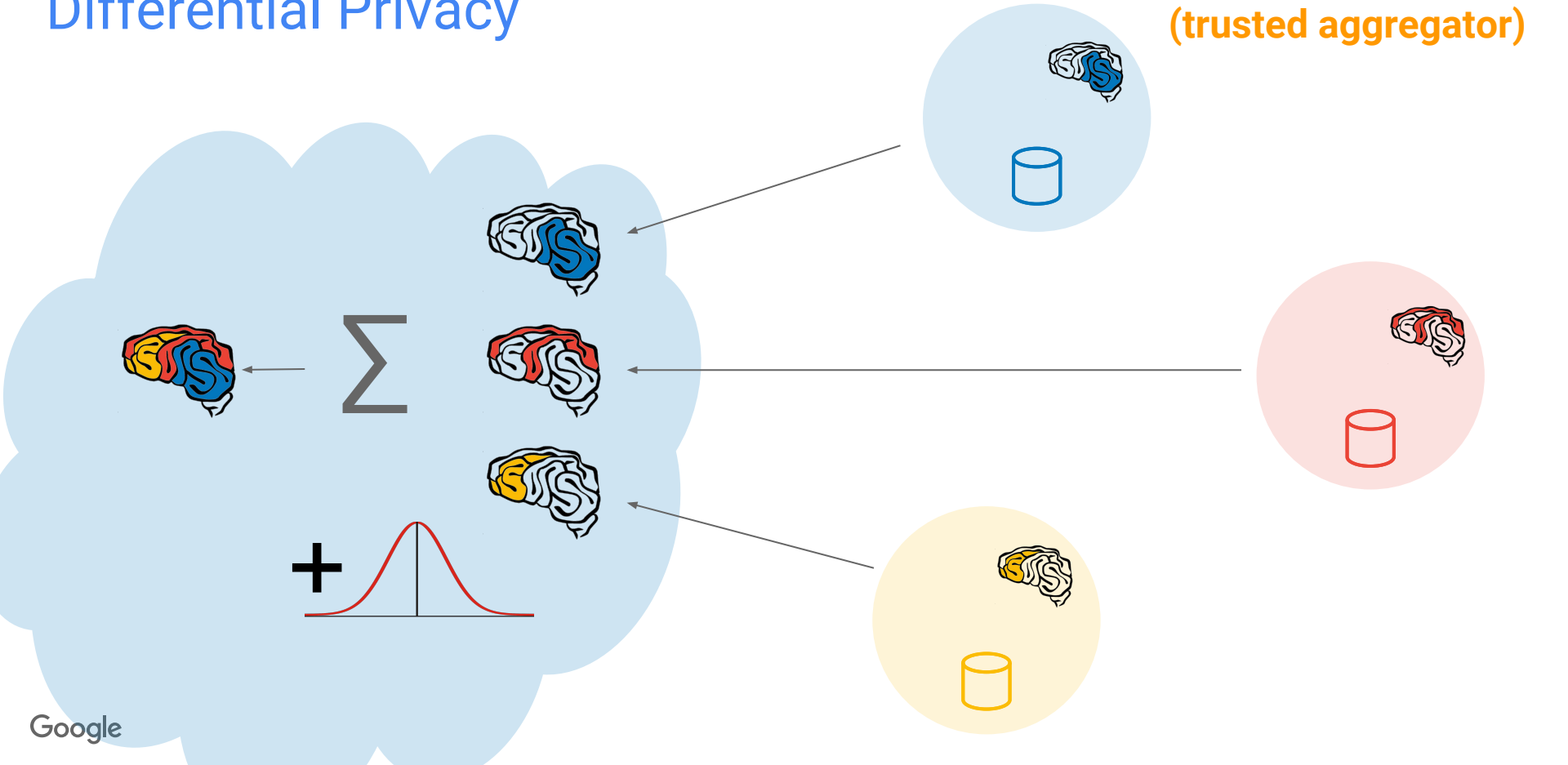
Baseline Training
users per round = 100
17.5% accuracy in 4120 rounds

(1.152, 1e-9) DP Training
$\mathbb{E}$[users per round] = 5k
17.5% accuracy in 5000 rounds

Private training achieves **equal accuracy**, but using **60x more computation**.

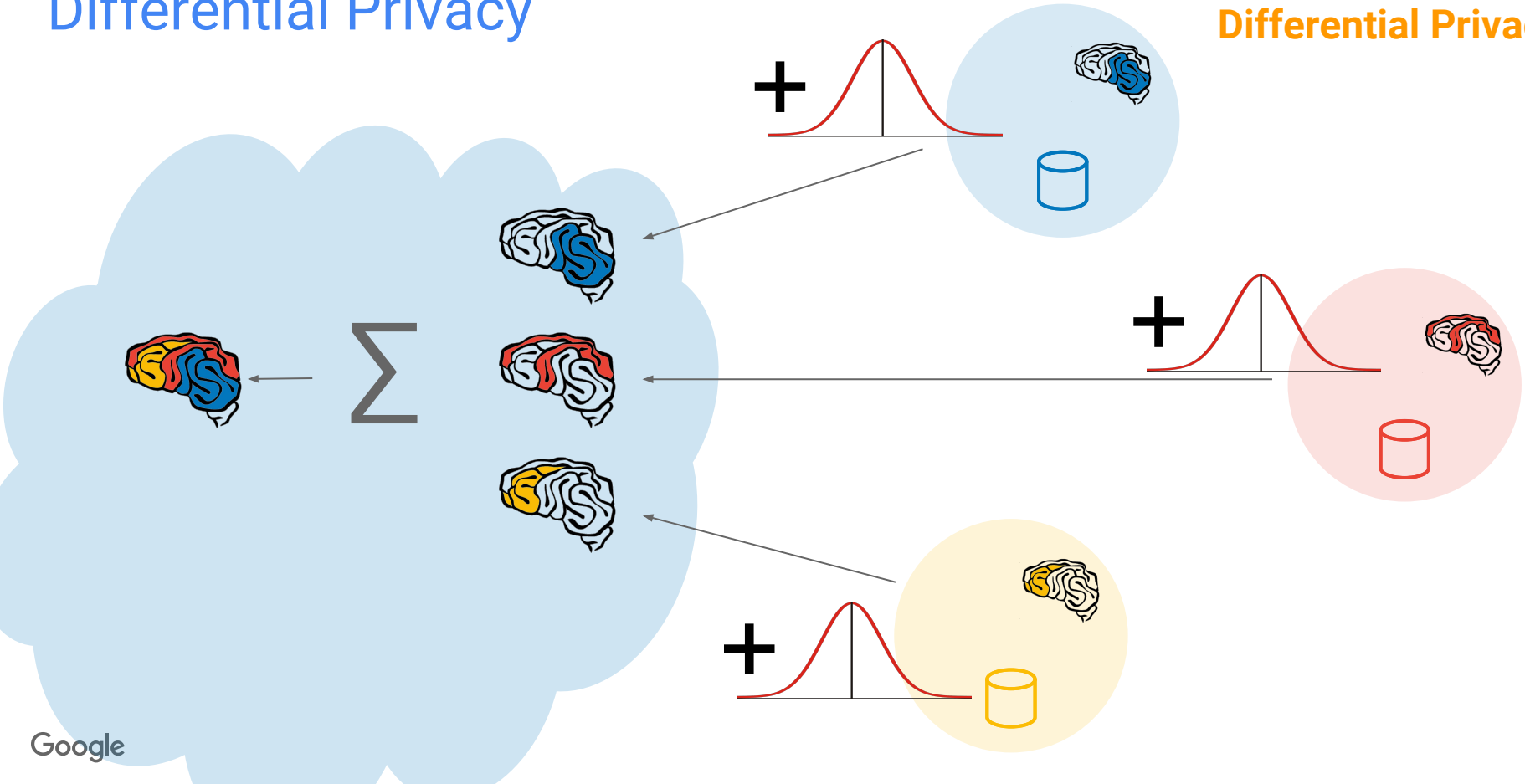
McMahan, Ramage, Talwar, Zhang. **Learning Differentially Private Recurrent Language Models**.

Differential Privacy



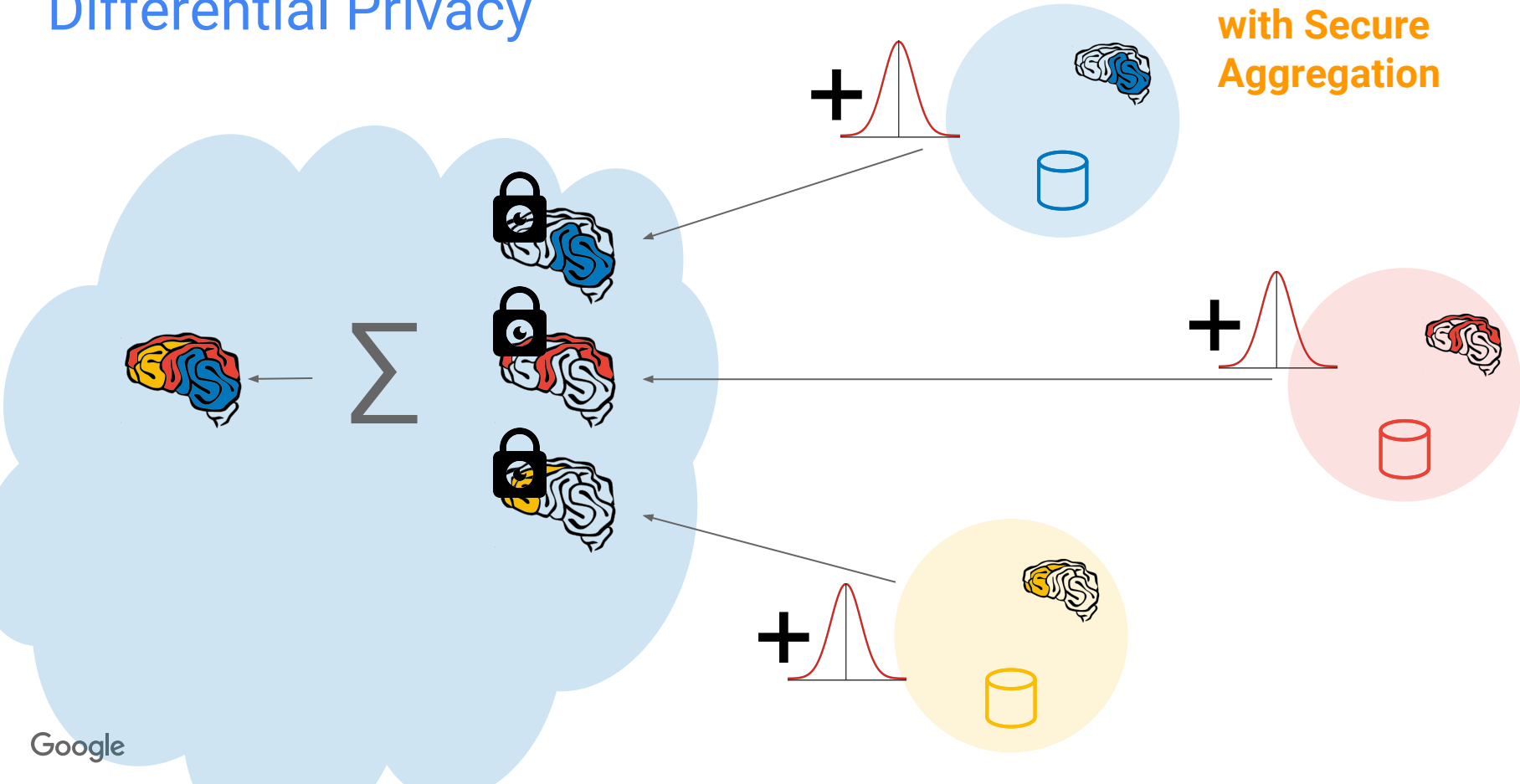
Differential Privacy

Local Differential Privacy



Differential Privacy

Differential Privacy with Secure Aggregation



Inherently Complementary Techniques

Federated Learning (FL)

- Touching each user's data infrequently is good for user-level DP guarantees
- FedAvg updates are simple averages, making DP analysis easier and allowing SA
- Averaging over many users allows high CU without losing too much signal

Differential Privacy (DP)

- Local noise may have regularization effect for FL (open question)
- Requires bounding update norms, reducing error of quantization in CU
- Bounding the norm of individual users may mitigate abuse potential under SA (open)

Secure Aggregation (SA)

- Less local noise for DP because individual updates not observed
- Quantization (necessary for SA) also used by CU. (more efficient combination open question)

Compressed Updates (CU)

- Lower communication costs good for FL settings, especially with SA
- Structured noise of CU may complement or replace local noise for DP (open question)

Federated Learning

Privacy-Preserving Collaborative Machine Learning without Centralized Training Data

Jakub Konečný / konkey@google.com

Google

On-device learning has many advantages

Federated Learning:

training a shared global model,
from a federation of participating devices
which maintain control of their own data,
with the facilitation of a central server.

Federated Learning is Practical

FederatedAveraging often converges quickly
(in terms of communication rounds)

Inherently Complementary Techniques

Federated Learning
Differential Privacy
Secure Aggregation
Compressed Updates

1. Why Federated Learning?
2. FL vs. other distributed optimization
3. FL Algorithms
4. Minimizing Communication
5. Secure Aggregation
6. Differential Privacy