Optimization

for Machine Learning in Practice II

Martin Jaggi

EPFL

Machine Learning and Optimization Laboratory

mlo.epfl.ch

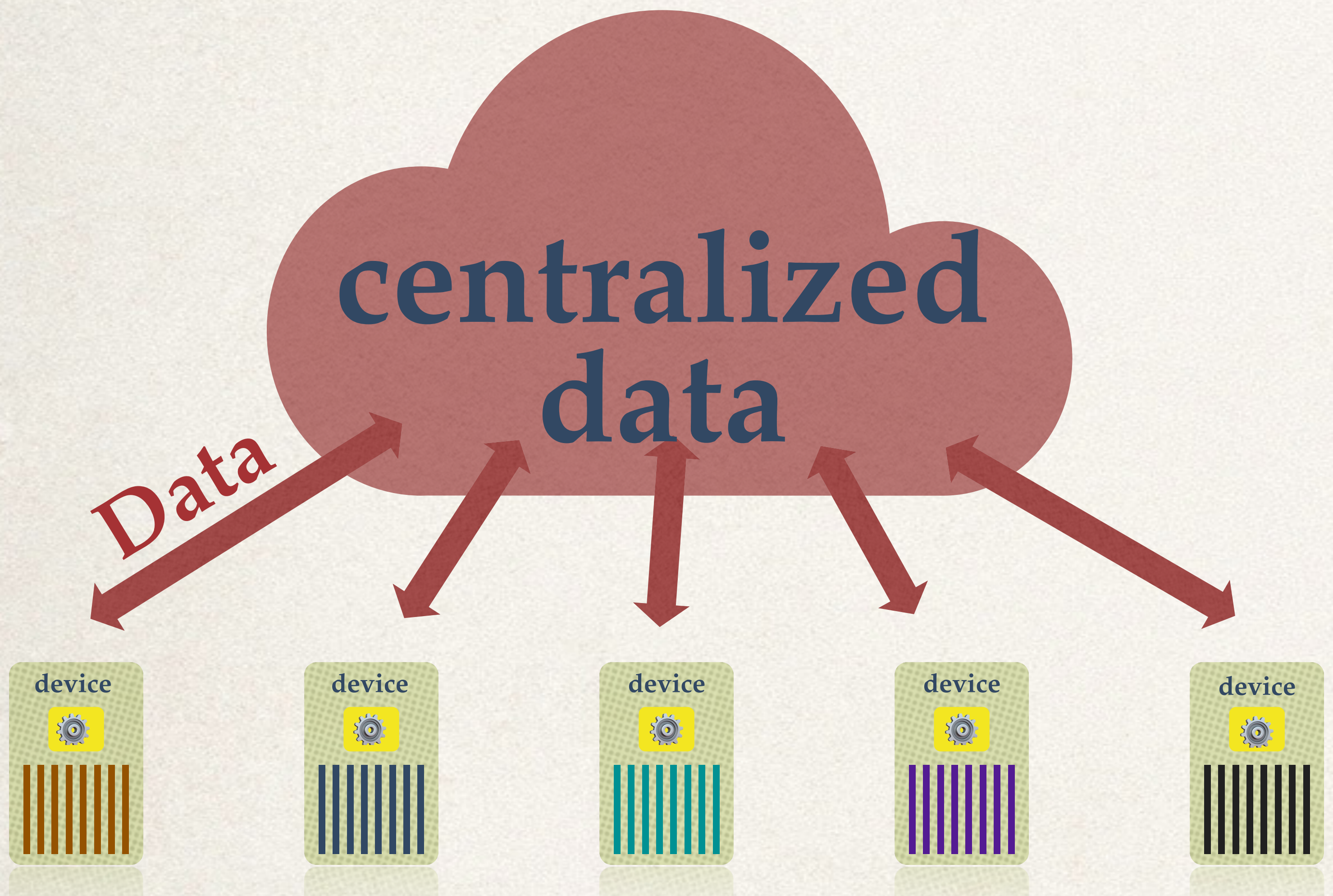
2

Collaborative Learning

Evolution

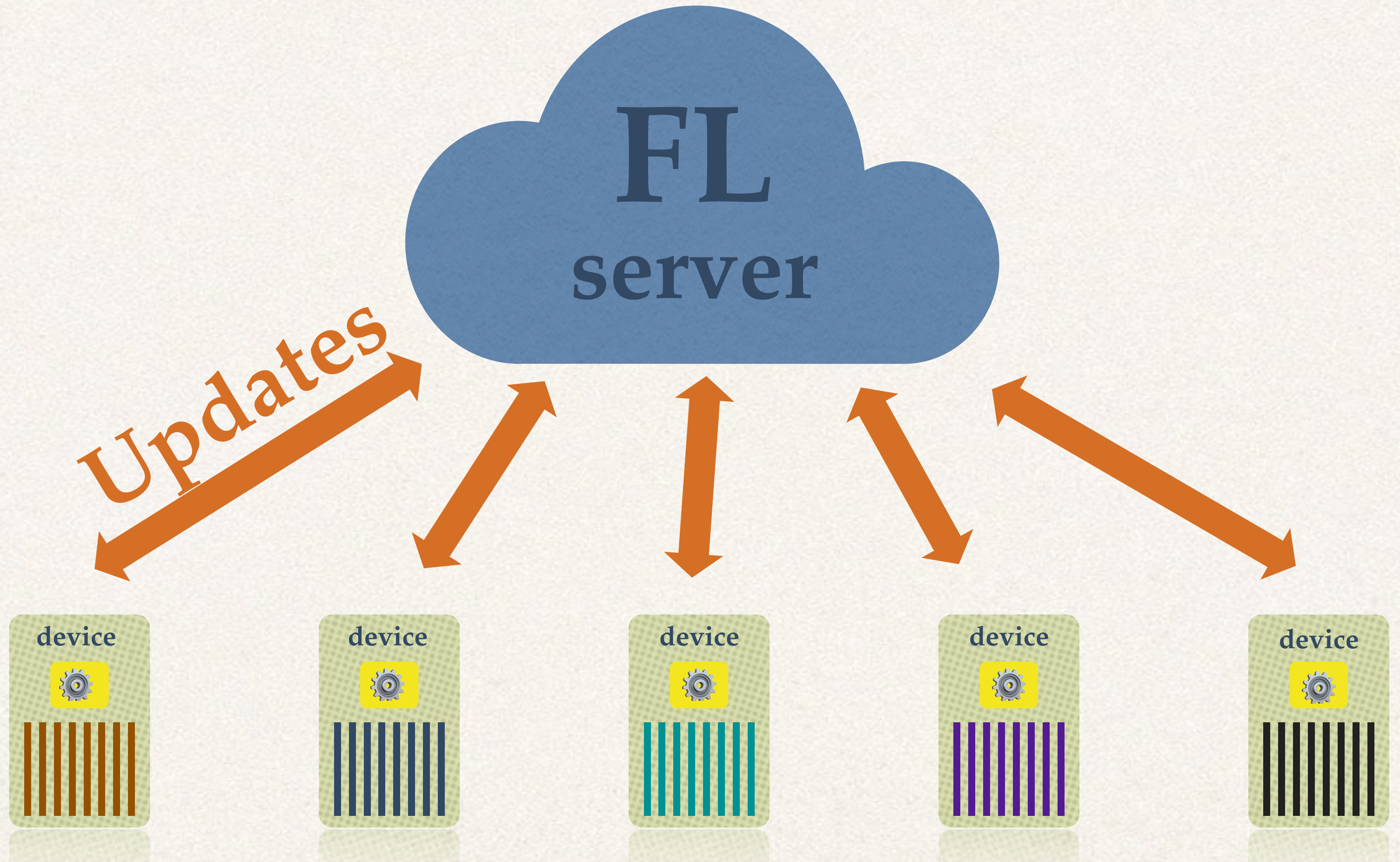
centralized

traditional, sharing data



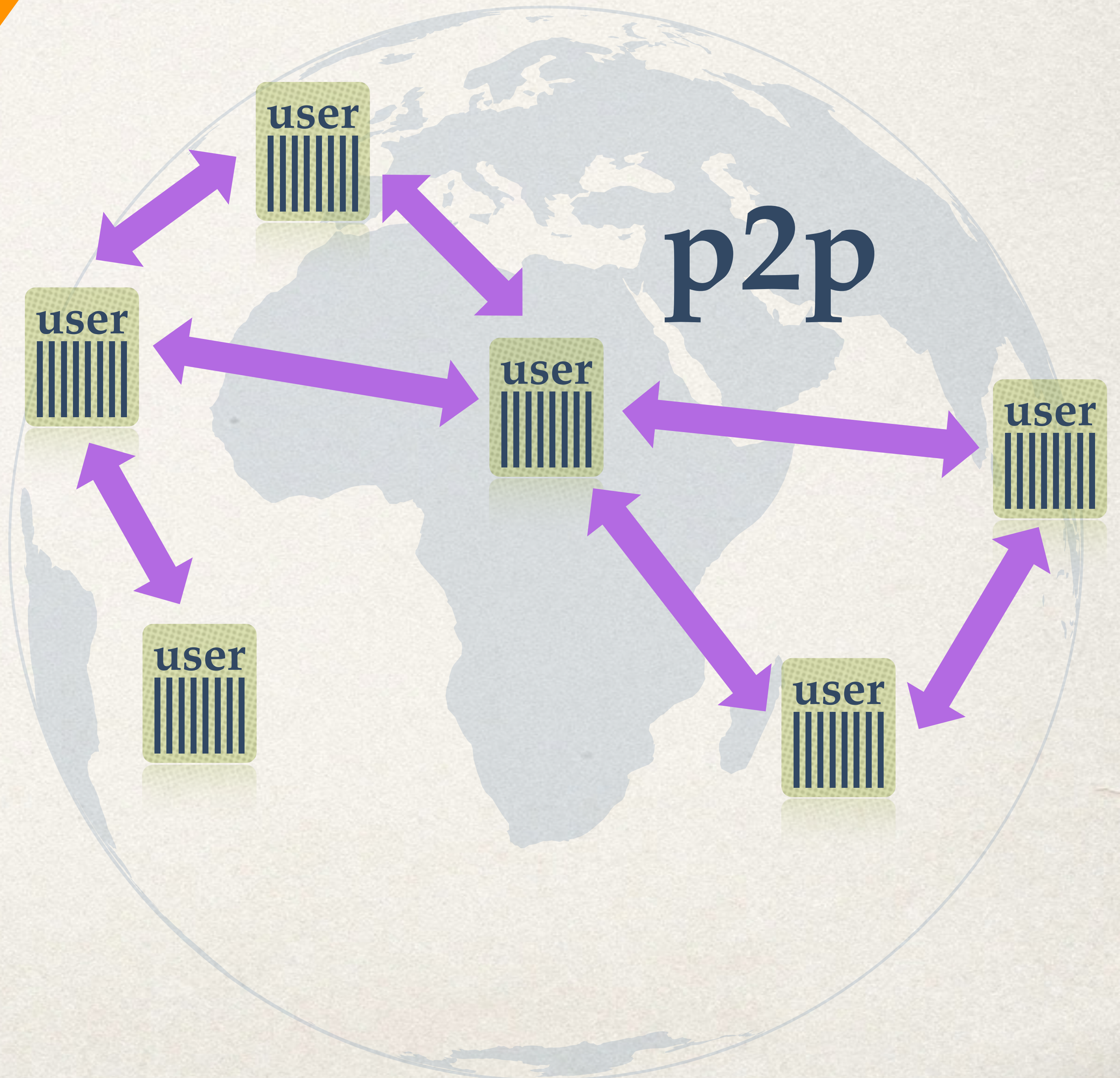
federated

sharing model updates

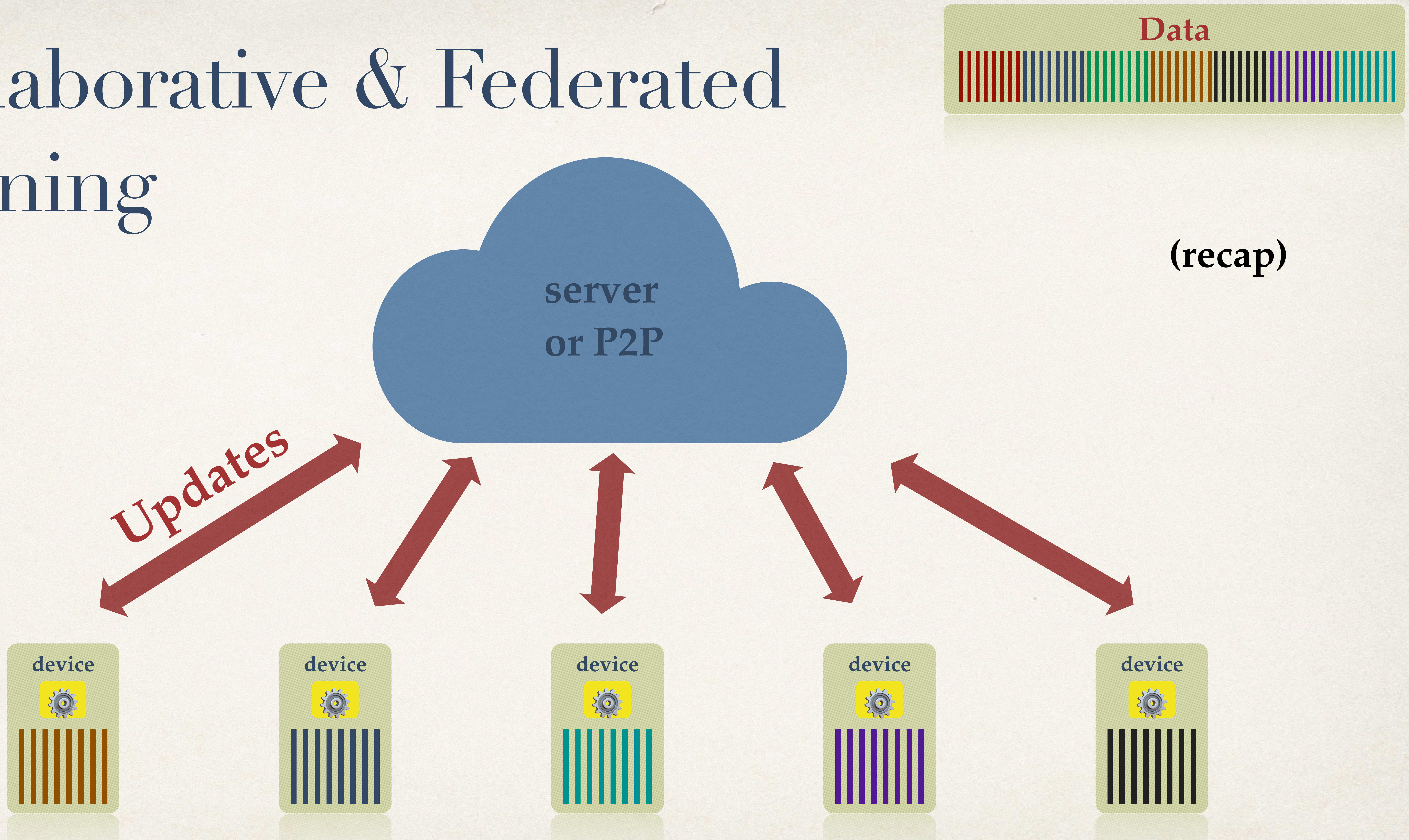


decentralized

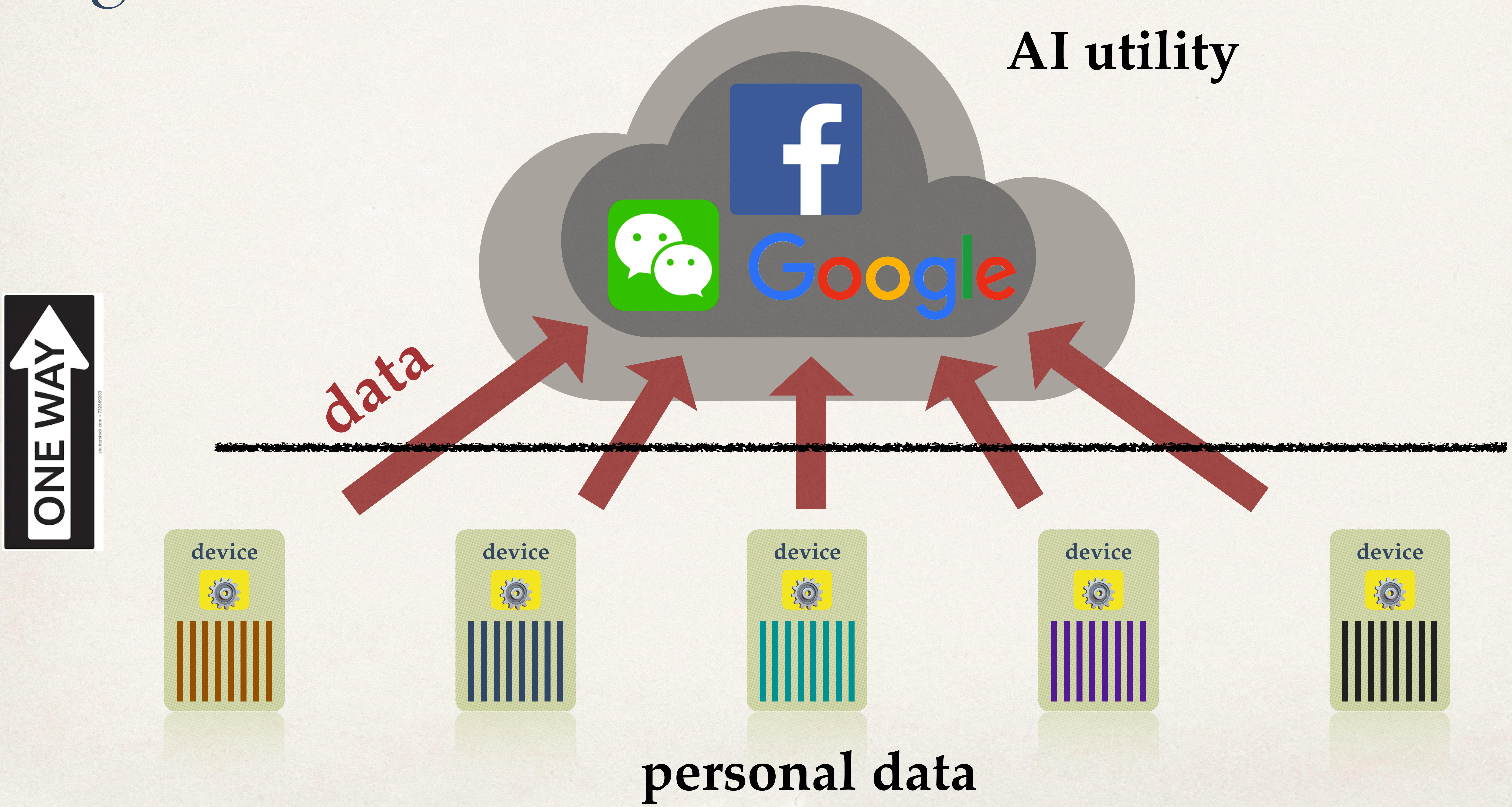
collaborative learning



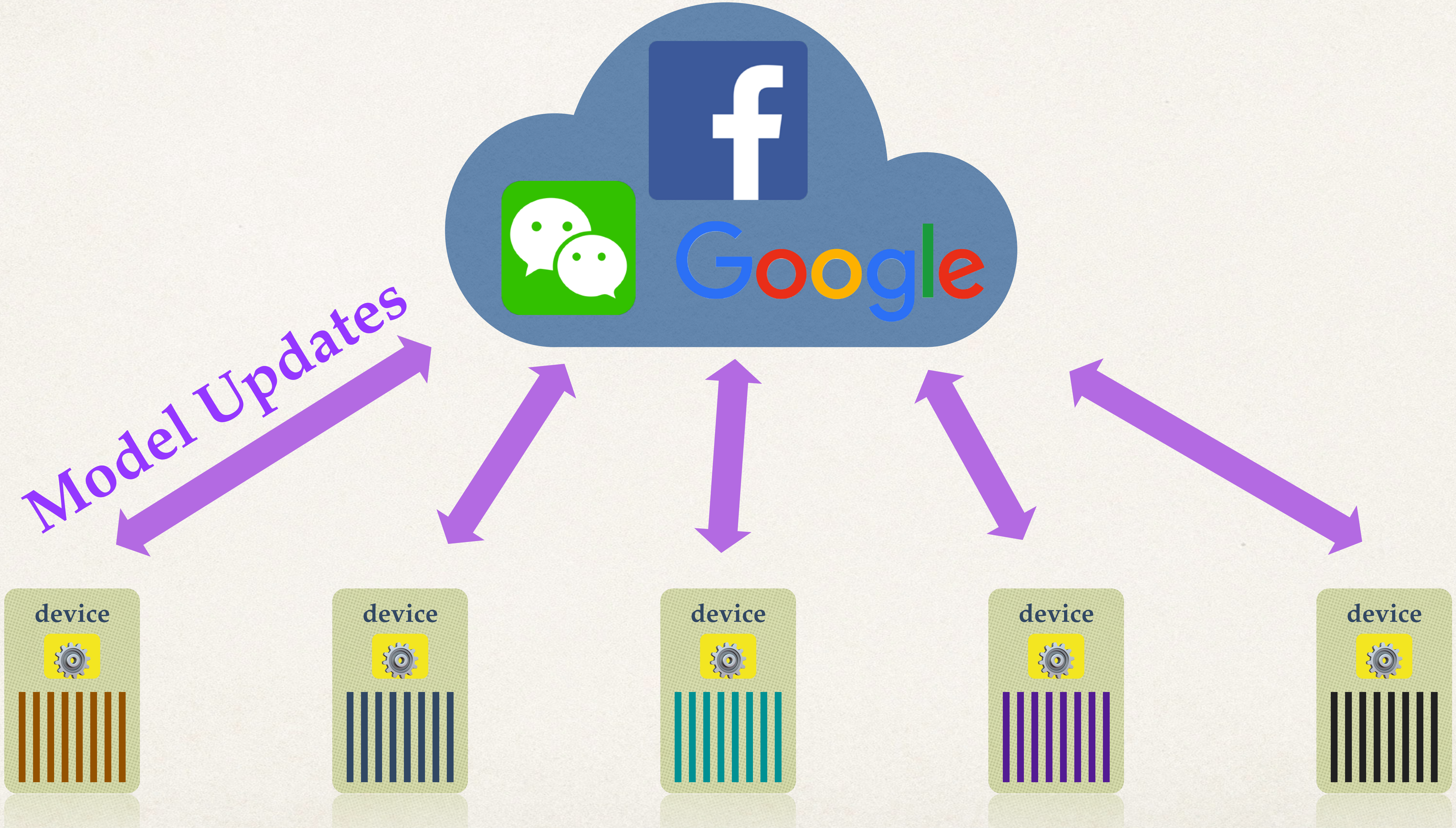
Collaborative & Federated Training



Big Picture

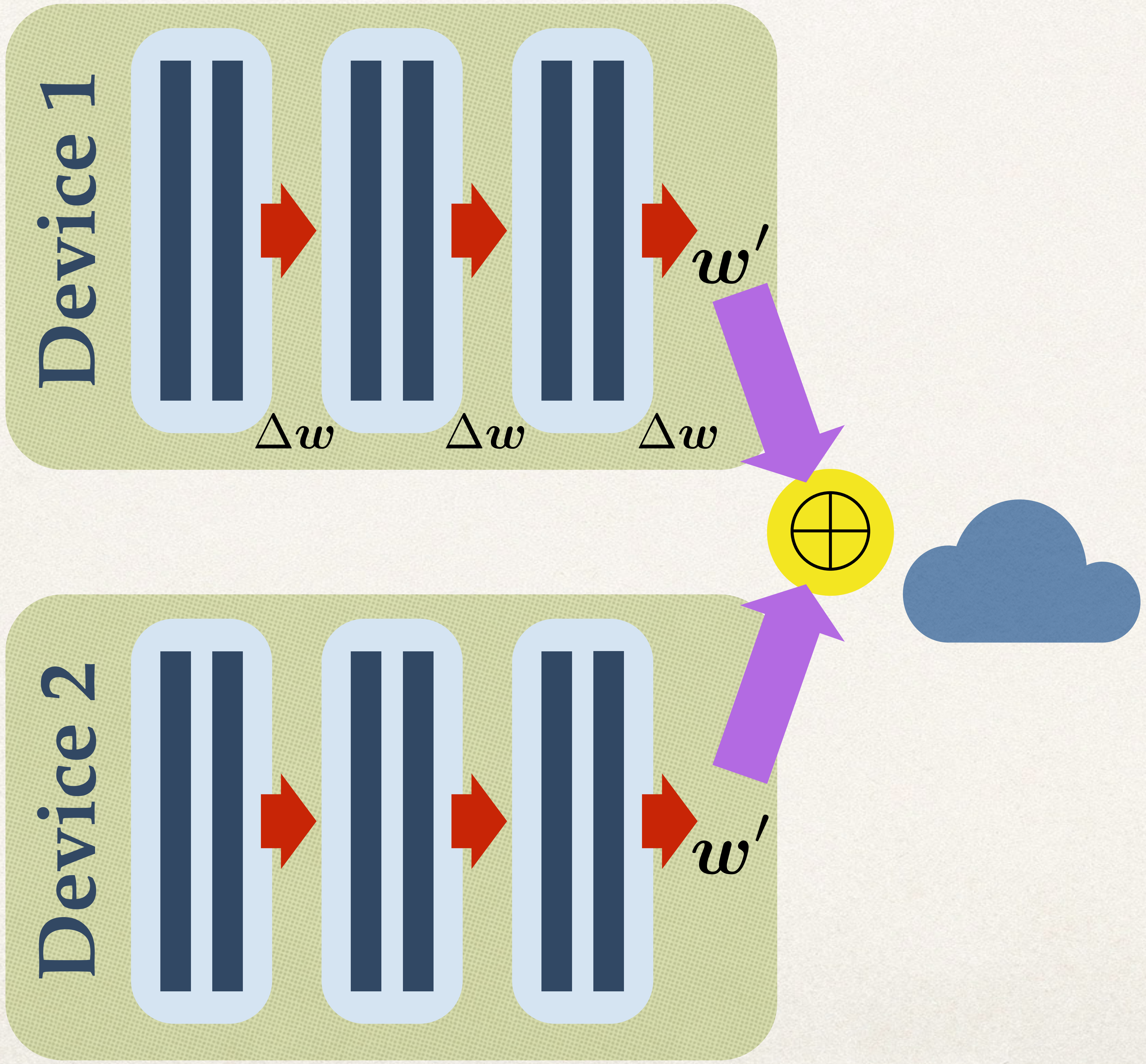

AI utility
Google
data
ONE WAY
device
device
device
device
device
personal data

2a Federated Learning



data never leaves device

2a Federated Learning



- Local SGD steps = "Federated averaging"
- Google Android Keyboard

Client drift

- **Federated Learning**

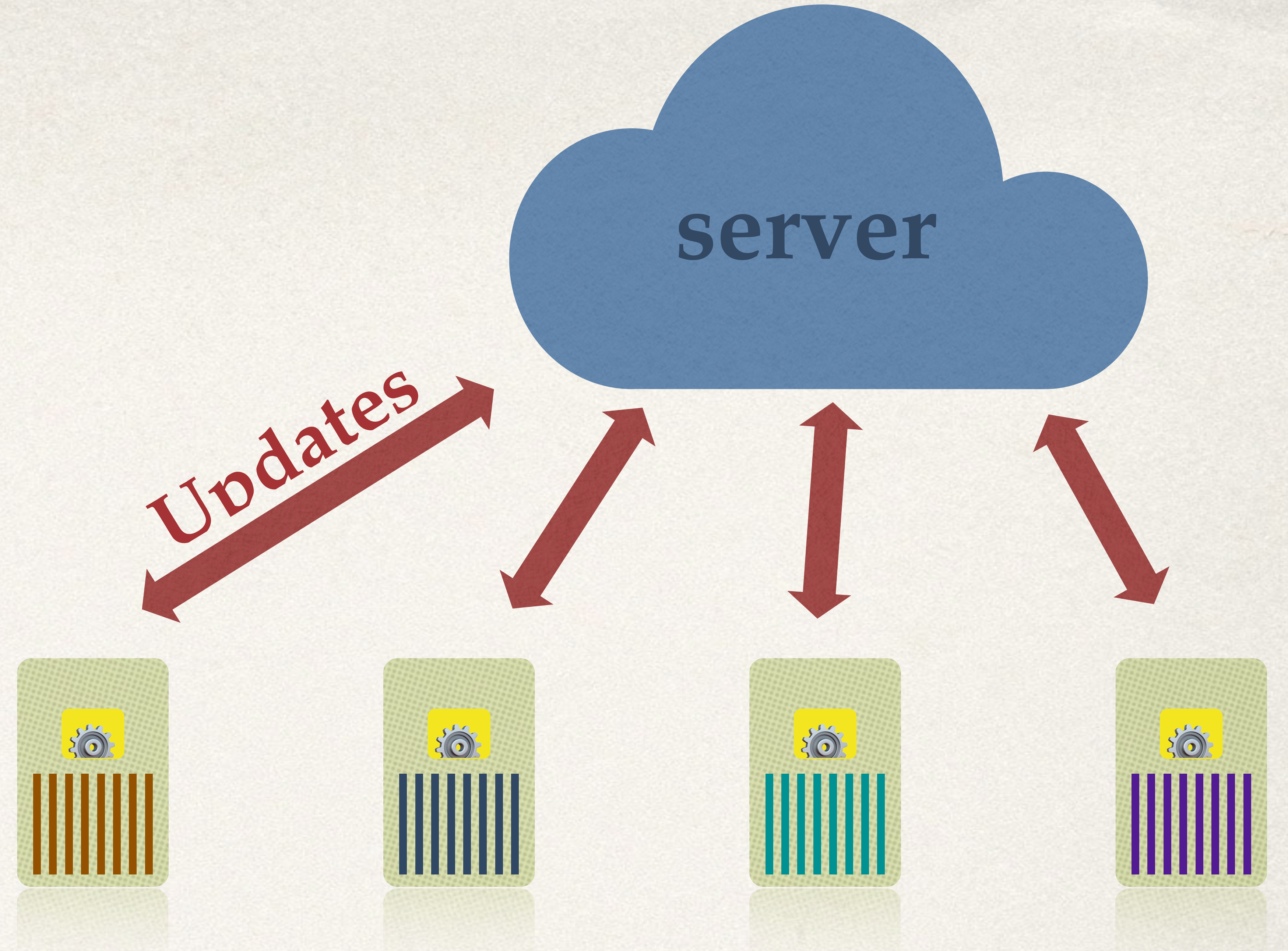
$$\min_{\boldsymbol{x}} \; \frac{1}{n}\sum_{i}^{n} f_i(\boldsymbol{x})$$

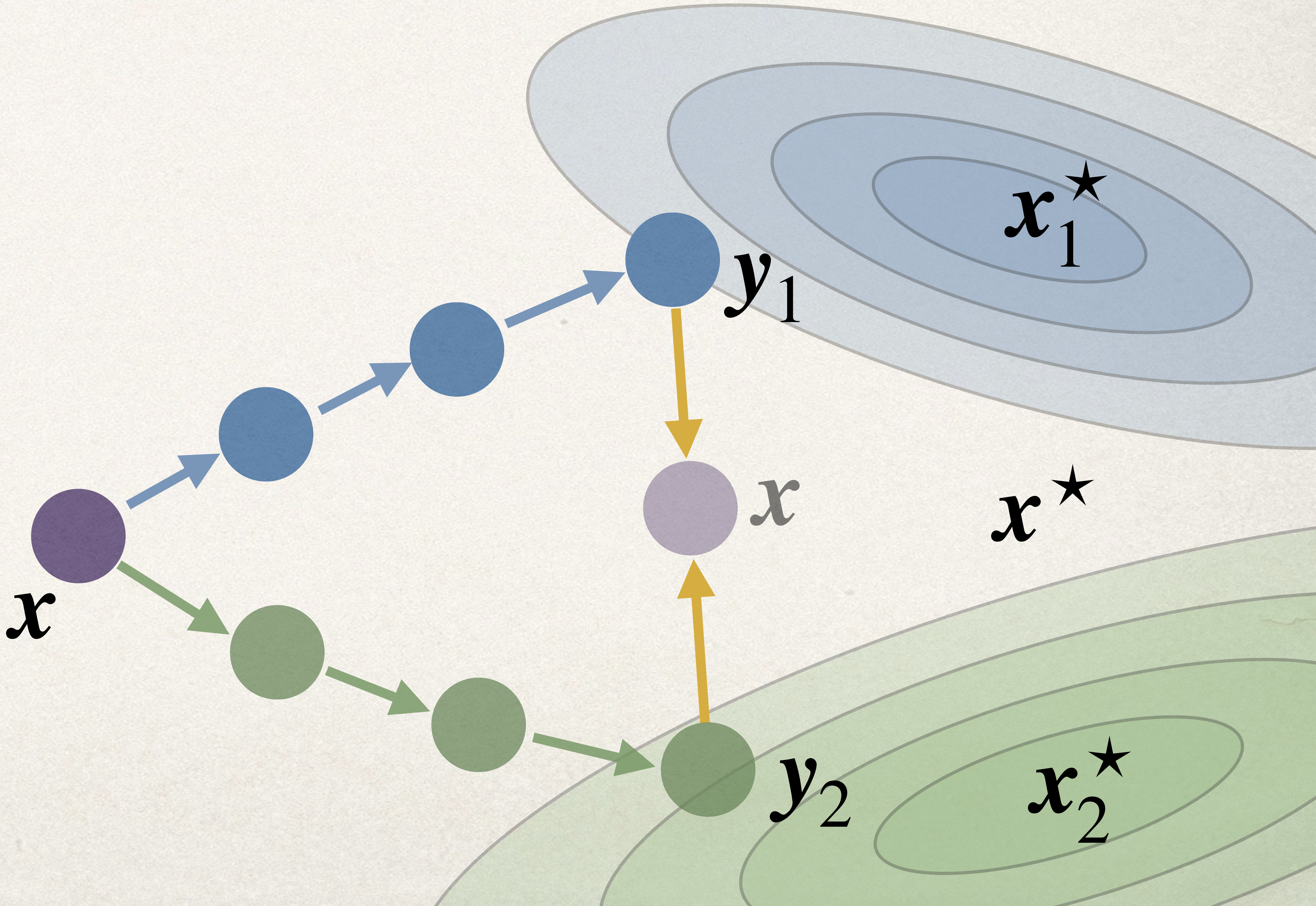
- **Fed Avg / Local SGD**

for some local steps

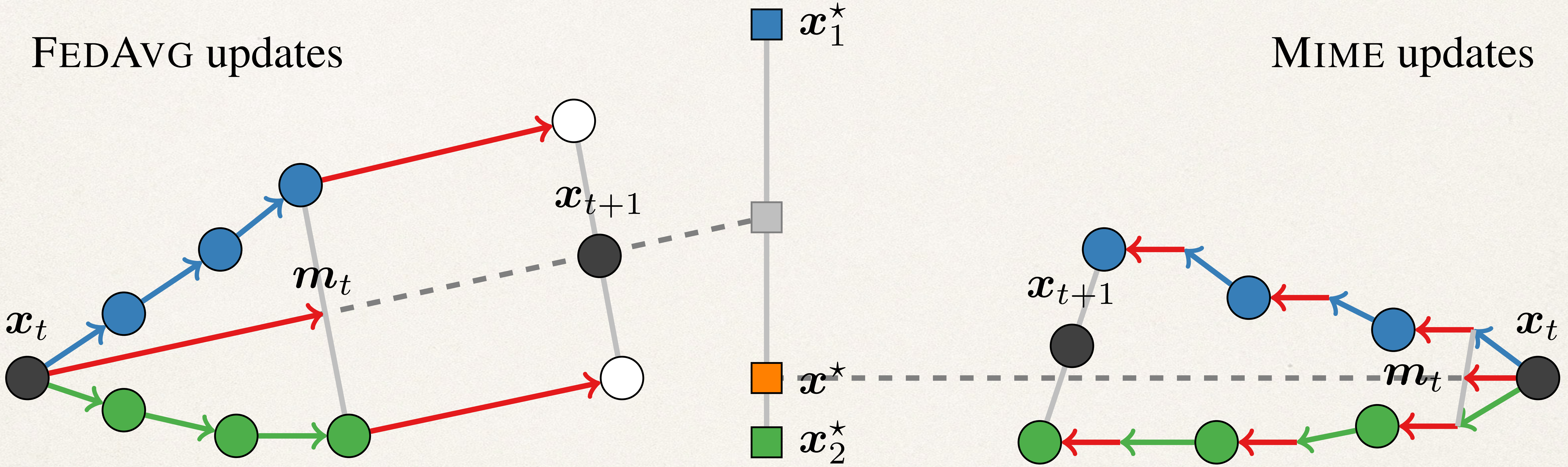
$$\boldsymbol{y}_i := \boldsymbol{y}_i - \eta \nabla f_i(\boldsymbol{y}_i)$$

$$\boldsymbol{x} := \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{y}_i \qquad \textit{(aggregation)}$$





Client drift



Mime algorithm framework

***for** some local steps*

$$\boldsymbol{y}_i := \boldsymbol{y}_i - \eta\Big((1-\beta)\,\nabla f_i(\boldsymbol{y}_i) + \beta\boldsymbol{m}\Big)$$

$$\boldsymbol{m} := (1-\beta)\,\nabla f(\boldsymbol{x}) + \beta\boldsymbol{m}$$

aggregated on server after each round

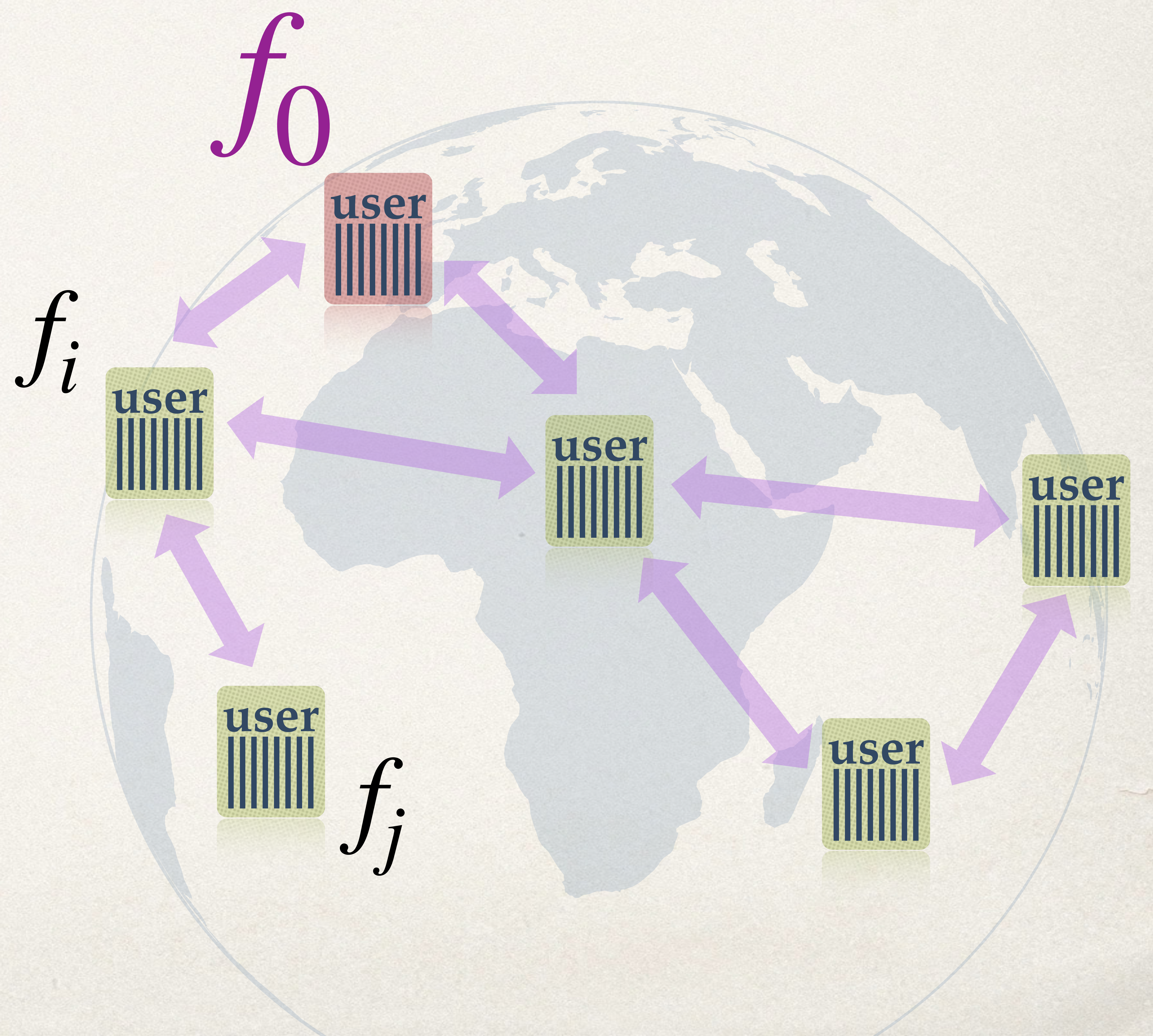
2b Federated vs Personalized Learning

- **Federated**

$$\min_{\boldsymbol{x}} \ \frac{1}{n}\sum_{i}^{n} f_i(\boldsymbol{x})$$

- **Collaborative / Personalized**

$$\min_{\boldsymbol{x}} \ f_0(\boldsymbol{x}) \quad \Bigg| \quad \begin{array}{l} \min_{\boldsymbol{x}} \ f_1(\boldsymbol{x}) \\ \\ \min_{\boldsymbol{x}} \ f_n(\boldsymbol{x}) \end{array}$$



2b Federated vs Personalized Learning

- **Federated**

$$\min_{\boldsymbol{x}} \ \frac{1}{n}\sum_{i}^{n} f_i(\boldsymbol{x})$$

- **Collaborative / Personalized**

$$\min_{\boldsymbol{x}} \ f_0(\boldsymbol{x}) \quad \Bigg| \quad \begin{matrix} \min_{\boldsymbol{x}} \ f_1(\boldsymbol{x}) \\ \\ \min_{\boldsymbol{x}} \ f_n(\boldsymbol{x}) \end{matrix}$$

- **Ordering of training**

  Set of active clients evolves (how?)

- **Clients = Tasks**

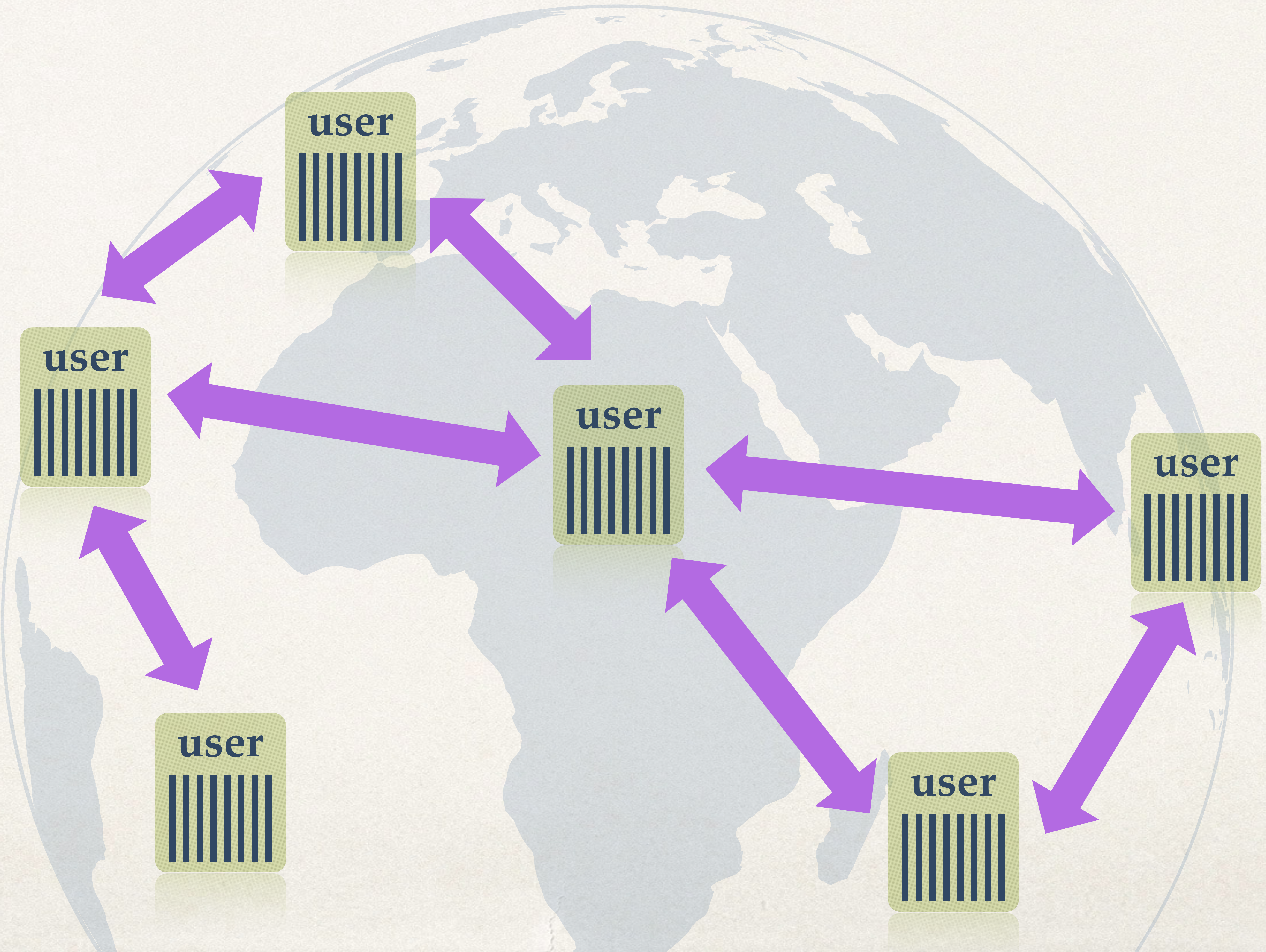
  Sequential fine-tuning

  Transfer learning,

  overparameterized models?

- **Train alone or collaborate?**

2c Decentralized Learning



Motivation

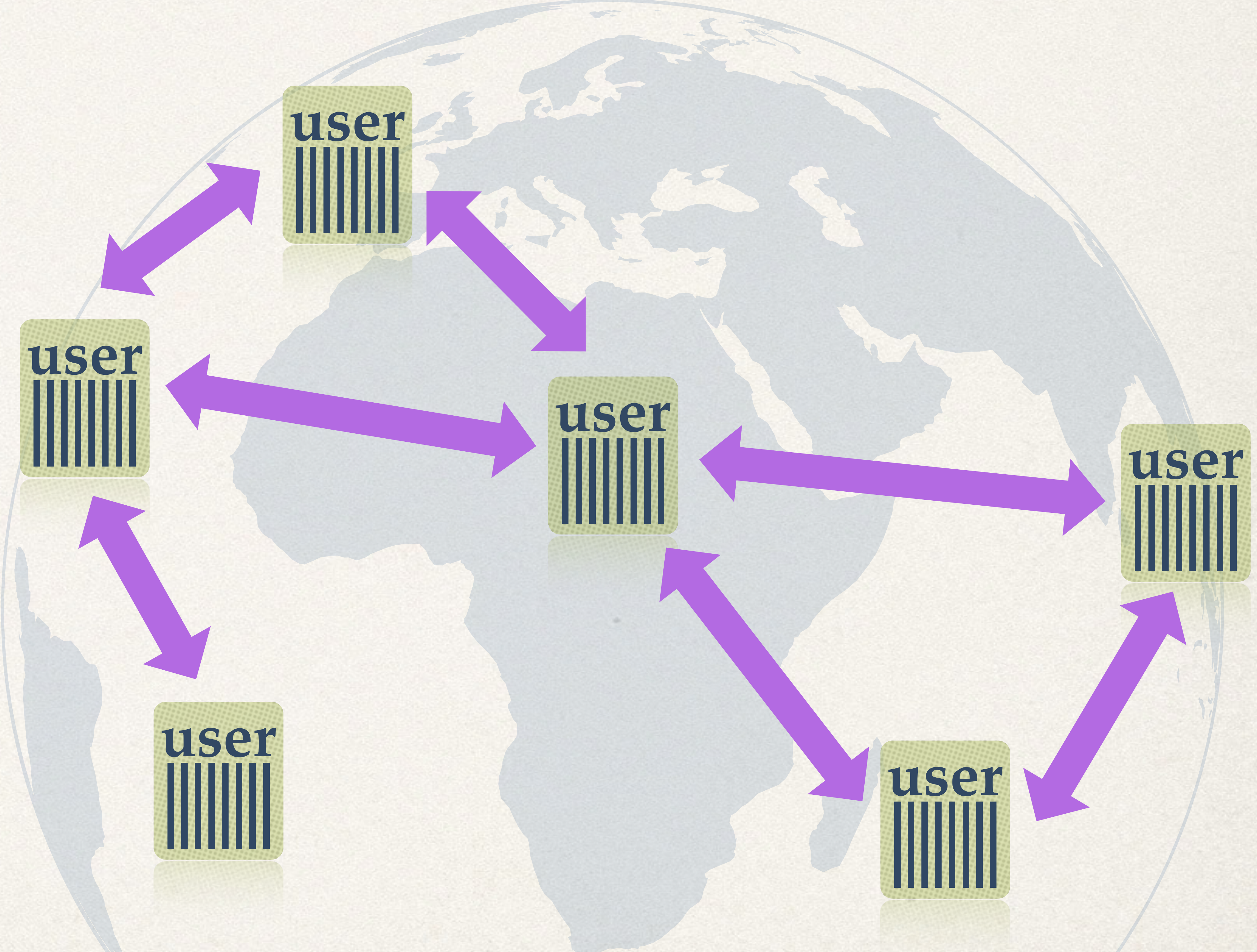
- **Applications:**

  any ML system with user data

  servers, devices, sensors, hospitals, ...

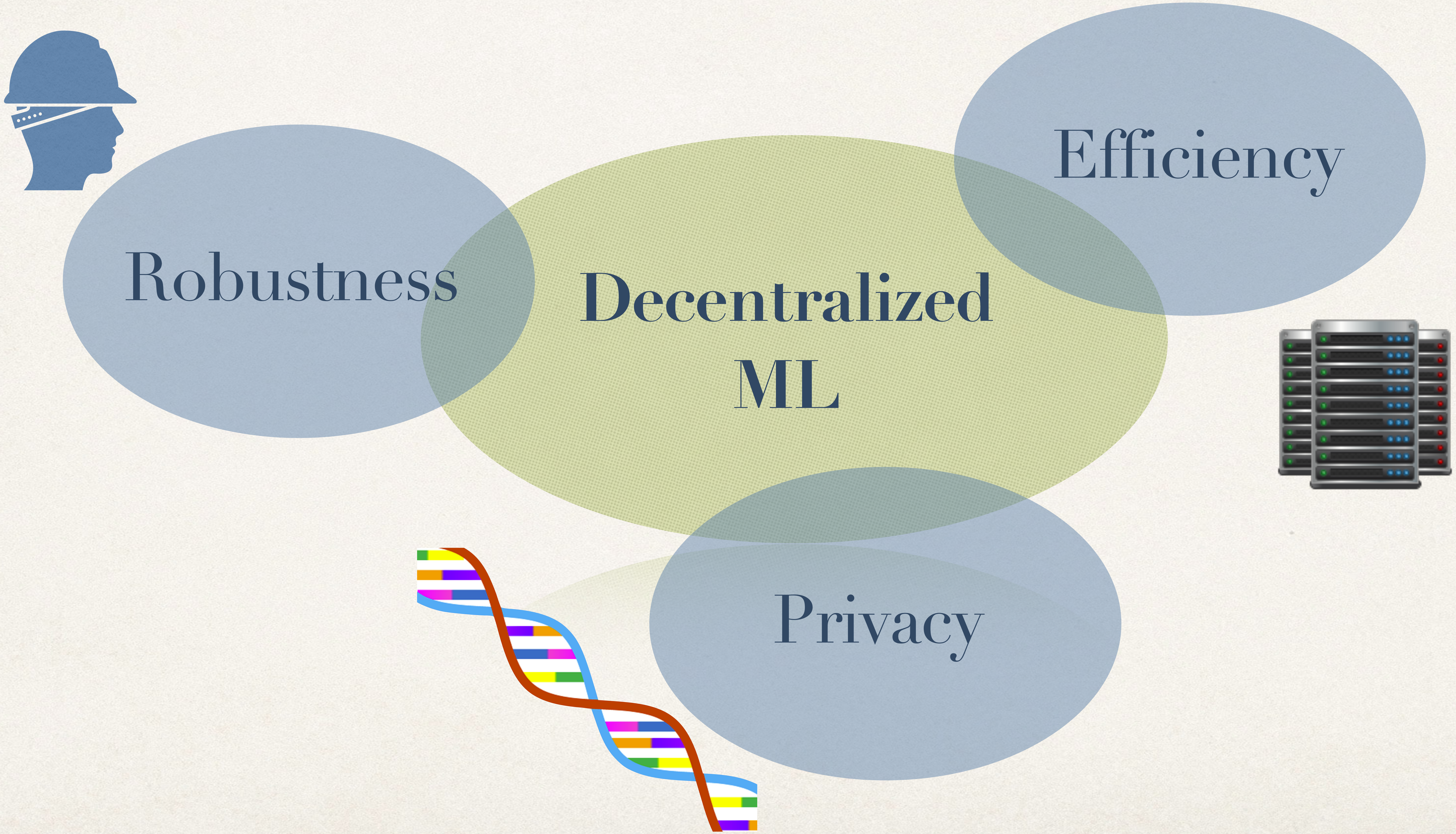
image source

- **Advantages:**

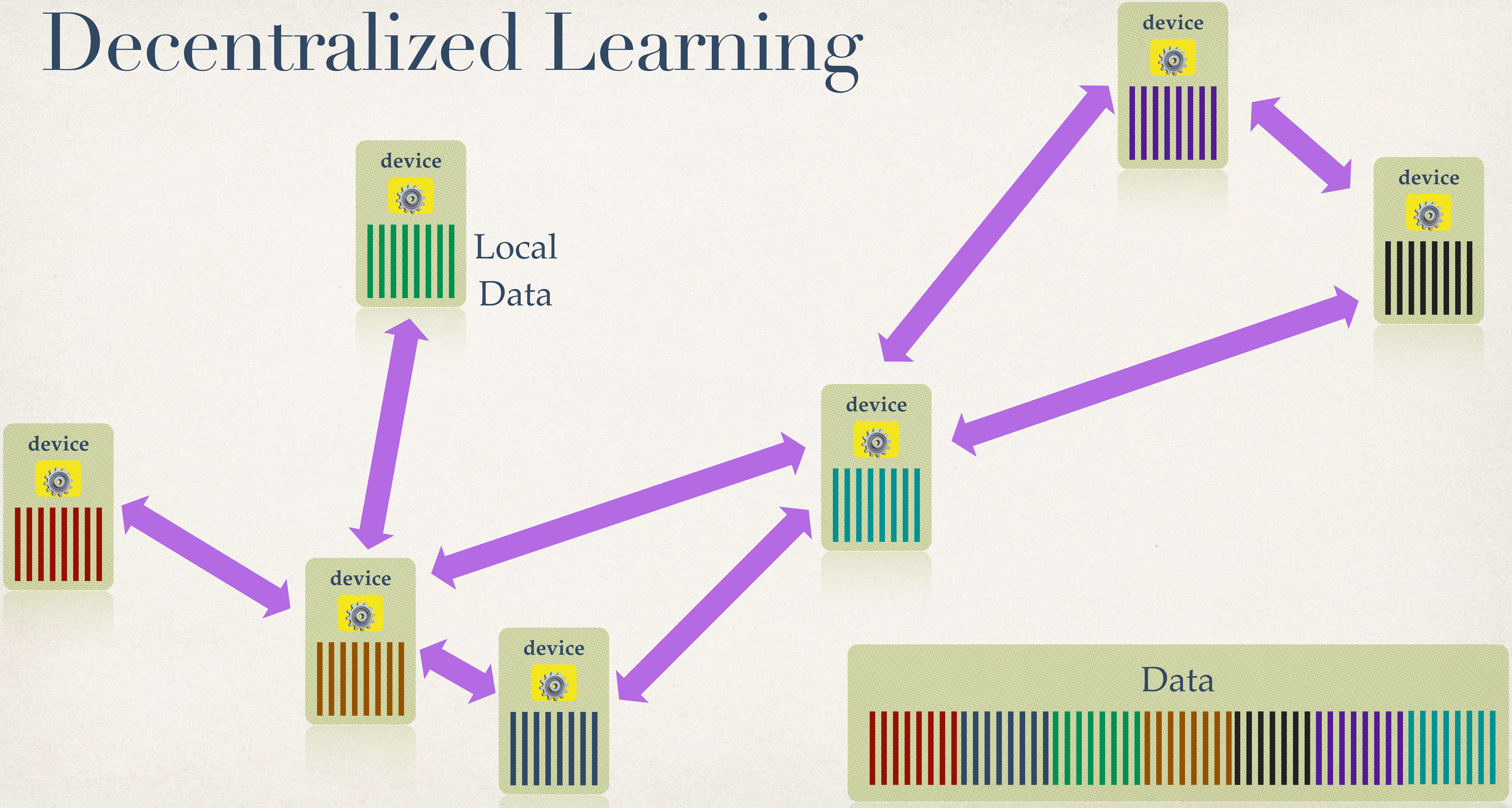


AI utility, control and privacy aligned with data ownership

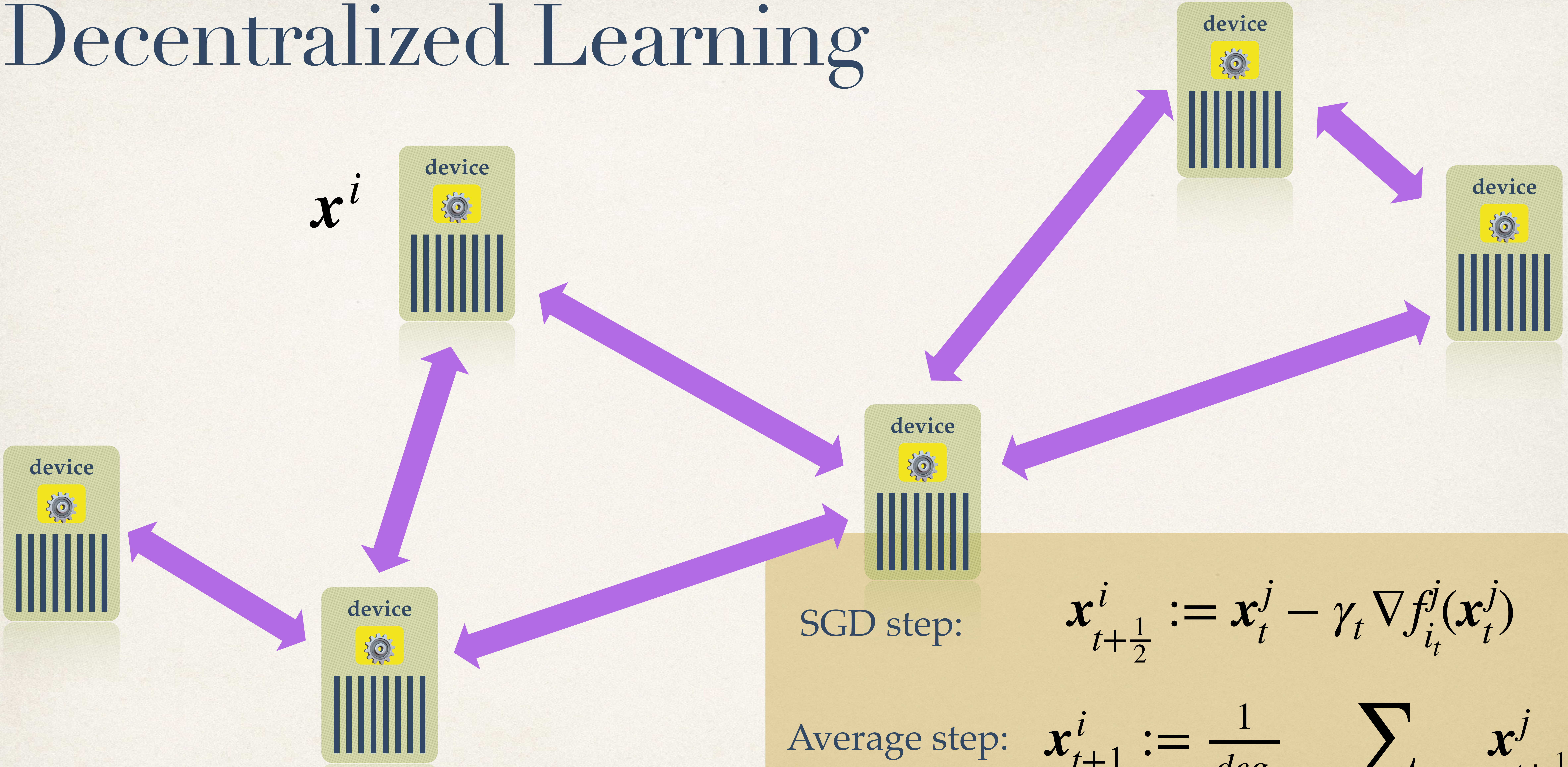
Required Building Blocks



Decentralized Learning



Decentralized Learning



SGD step: $$\boldsymbol{x}^i_{t+\frac{1}{2}} := \boldsymbol{x}^j_t - \gamma_t \nabla f^j_{i_t}(\boldsymbol{x}^j_t)$$

Average step: $$\boldsymbol{x}^i_{t+1} := \frac{1}{deg_i} \sum_{j:neighbours} \boldsymbol{x}^j_{t+\frac{1}{2}}$$

Communication Compression

- limited-bit precision vector

  e.g. 1-bit per entry reduces communication 32 times

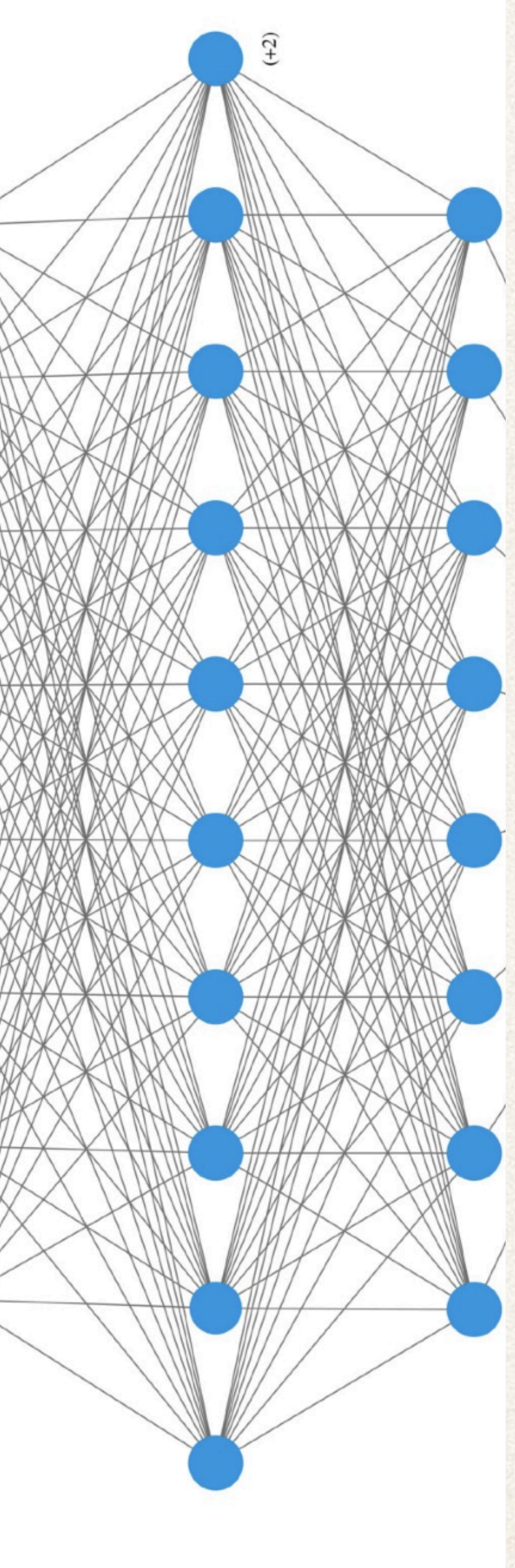
- random/top k% of all the entries

  e.g. k=0.1% reduces communication 1000 times

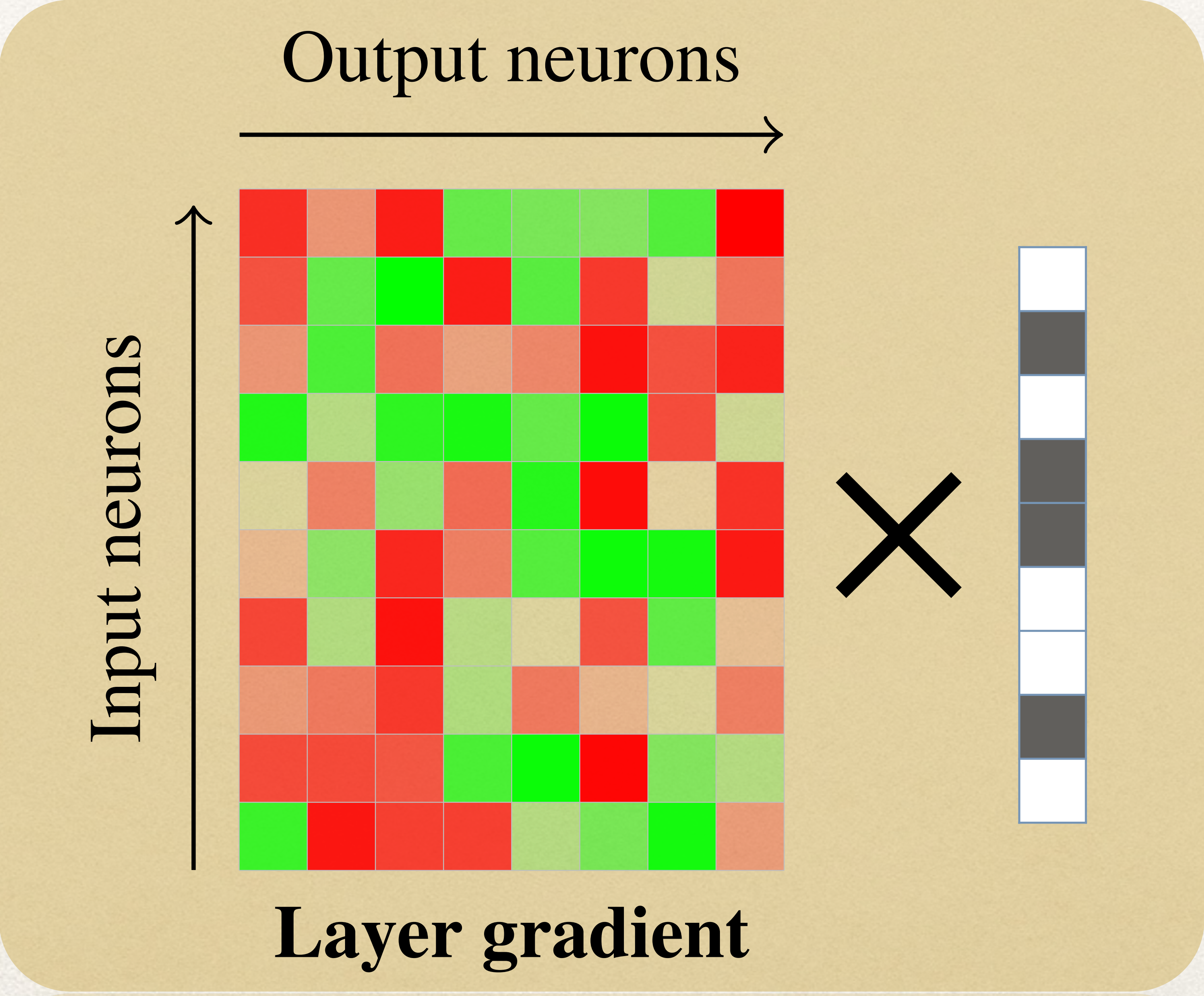
- low rank version of the gradient?

Low-Rank Communication Compression

- PowerSGD



backprop is fast:
linear time



fast compression?

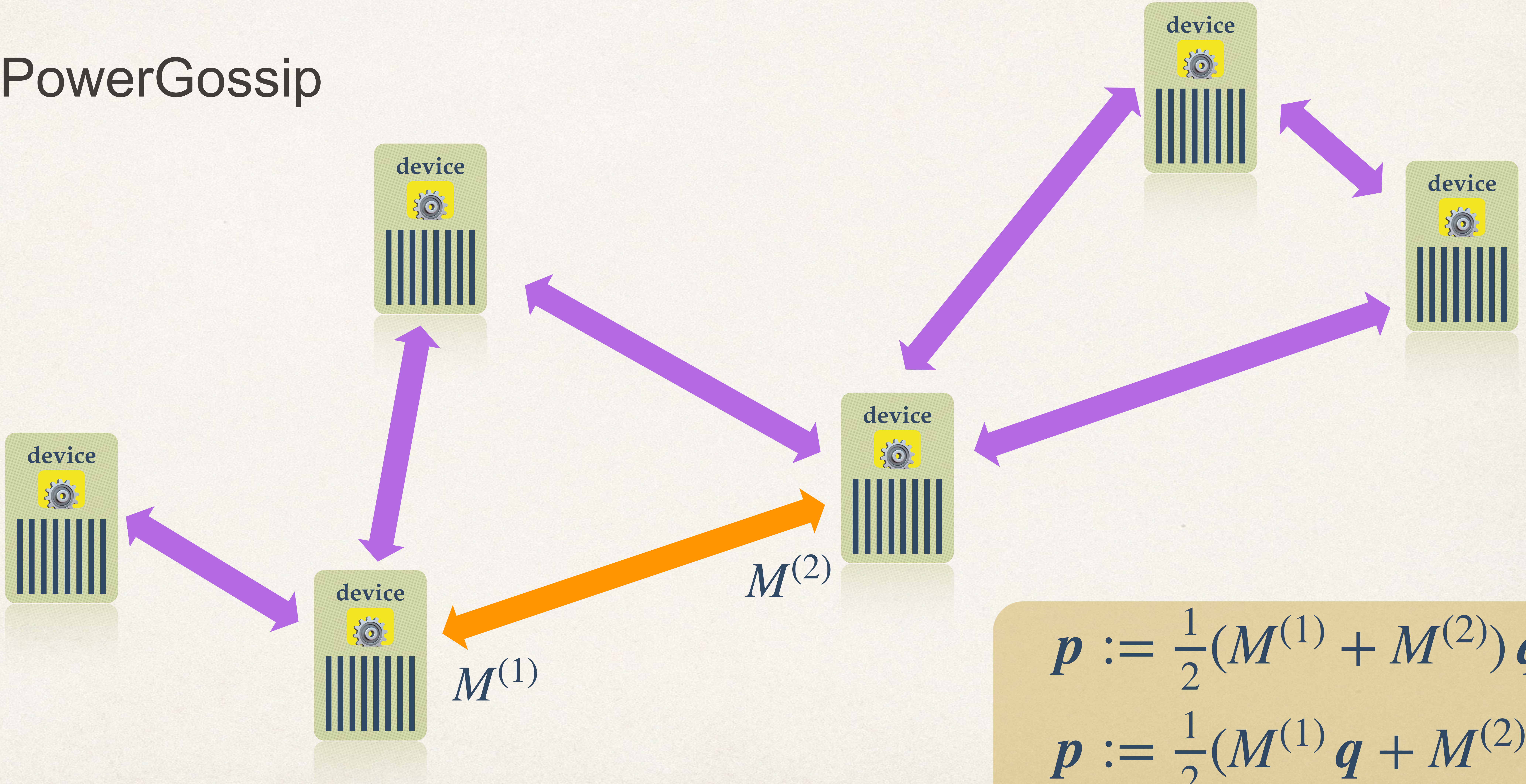
Fast power iterations

$$\boldsymbol{p} := G\boldsymbol{q}$$
$$\boldsymbol{q} := G^\top\boldsymbol{p}$$

$$\mathscr{C}(G) = \boldsymbol{p}\boldsymbol{q}^\top$$

Decentralized Learning with Compression

- PowerGossip



$$p := \frac{1}{2}(M^{(1)} + M^{(2)})\, q$$

$$p := \frac{1}{2}(M^{(1)} q + M^{(2)} q)$$

Building Blocks for Decentralized ML

- **Efficiency: Communication & Compute**

  on-device learning, Edge AI

  peer-to-peer communication

- **Privacy**

  data locality, leakage?, attacks?

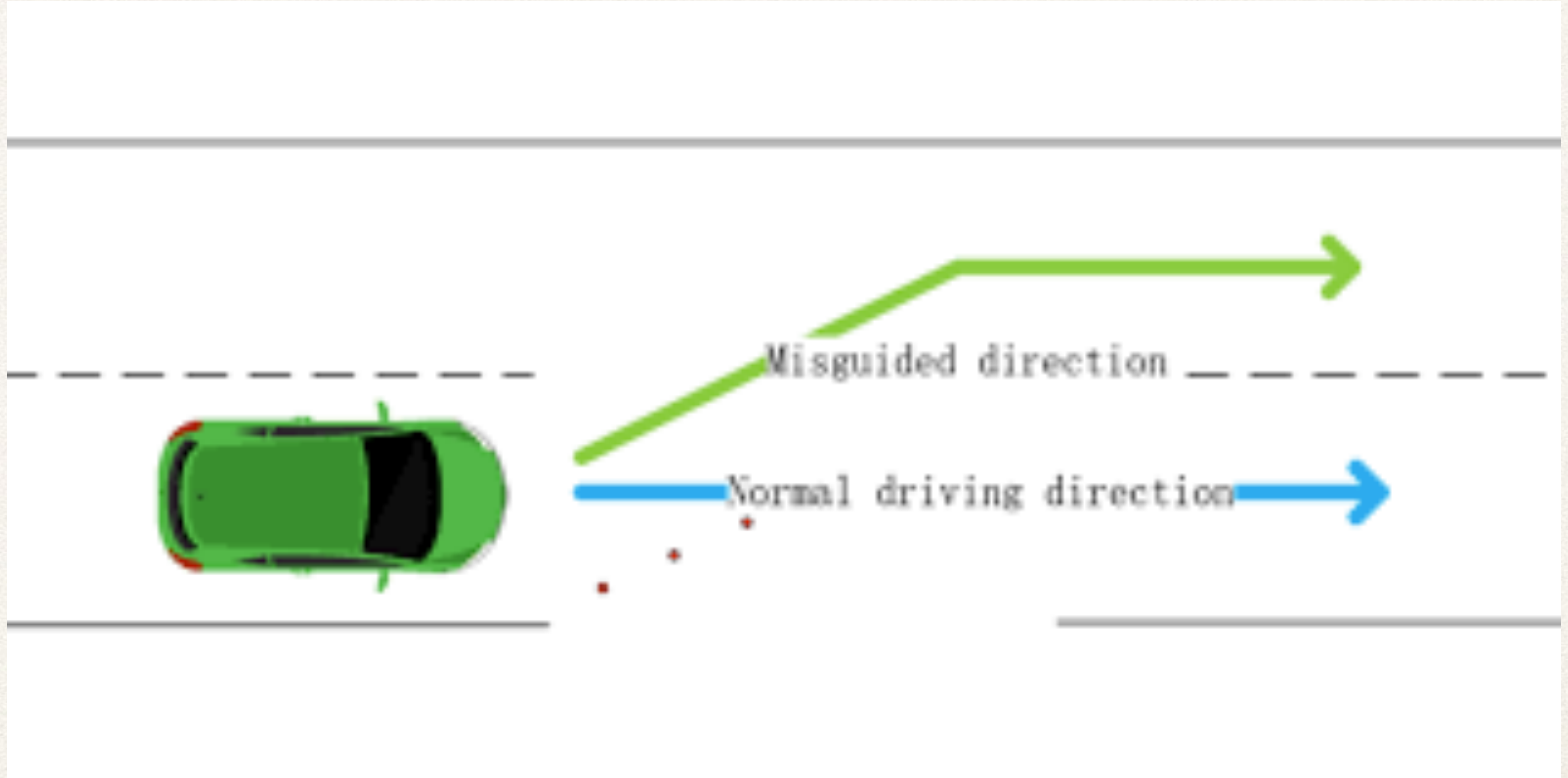
- **Robustness & Incentives**

  tolerate bad players, reward collaboration

3

Robustness

During Training and Inference

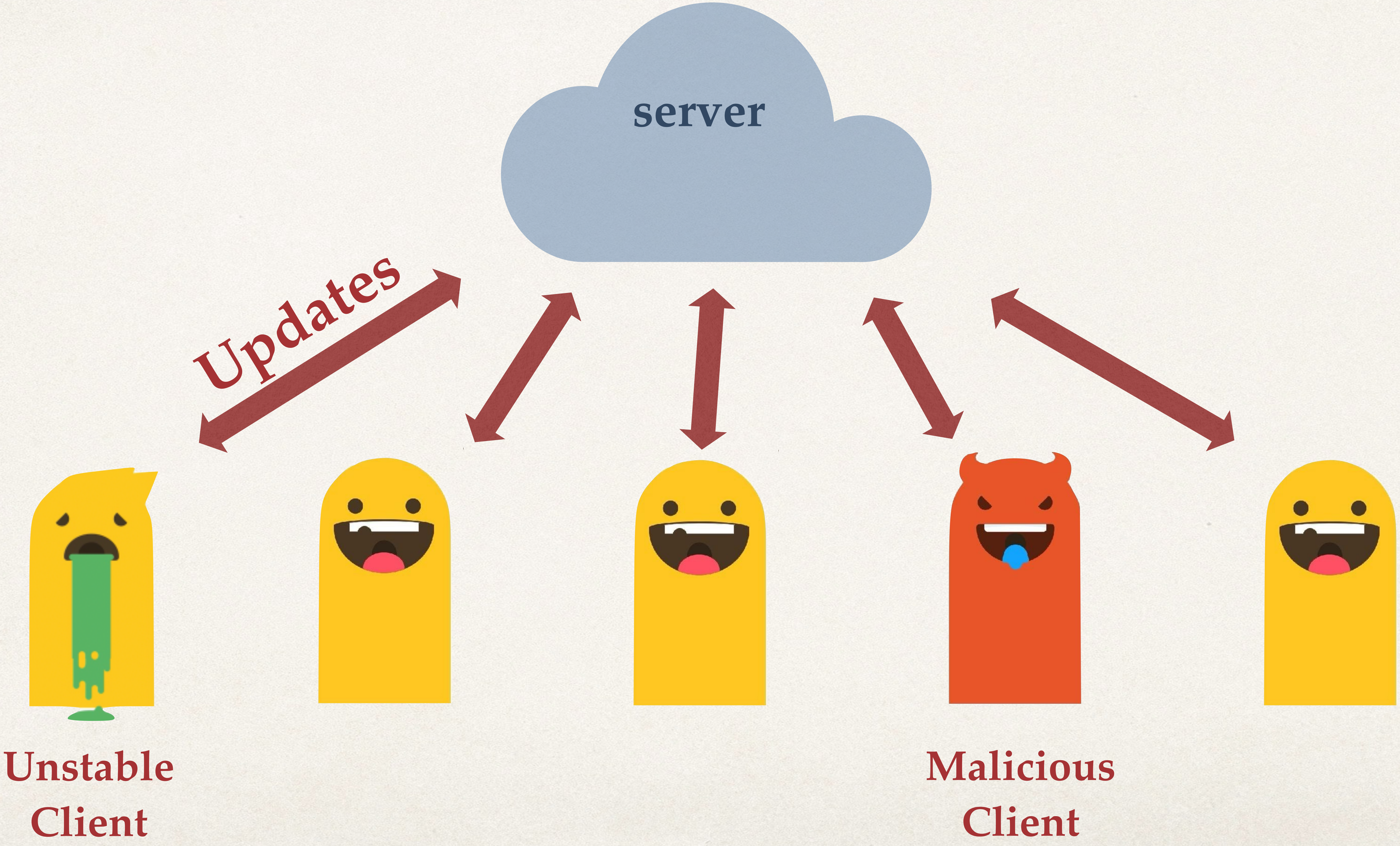


3a

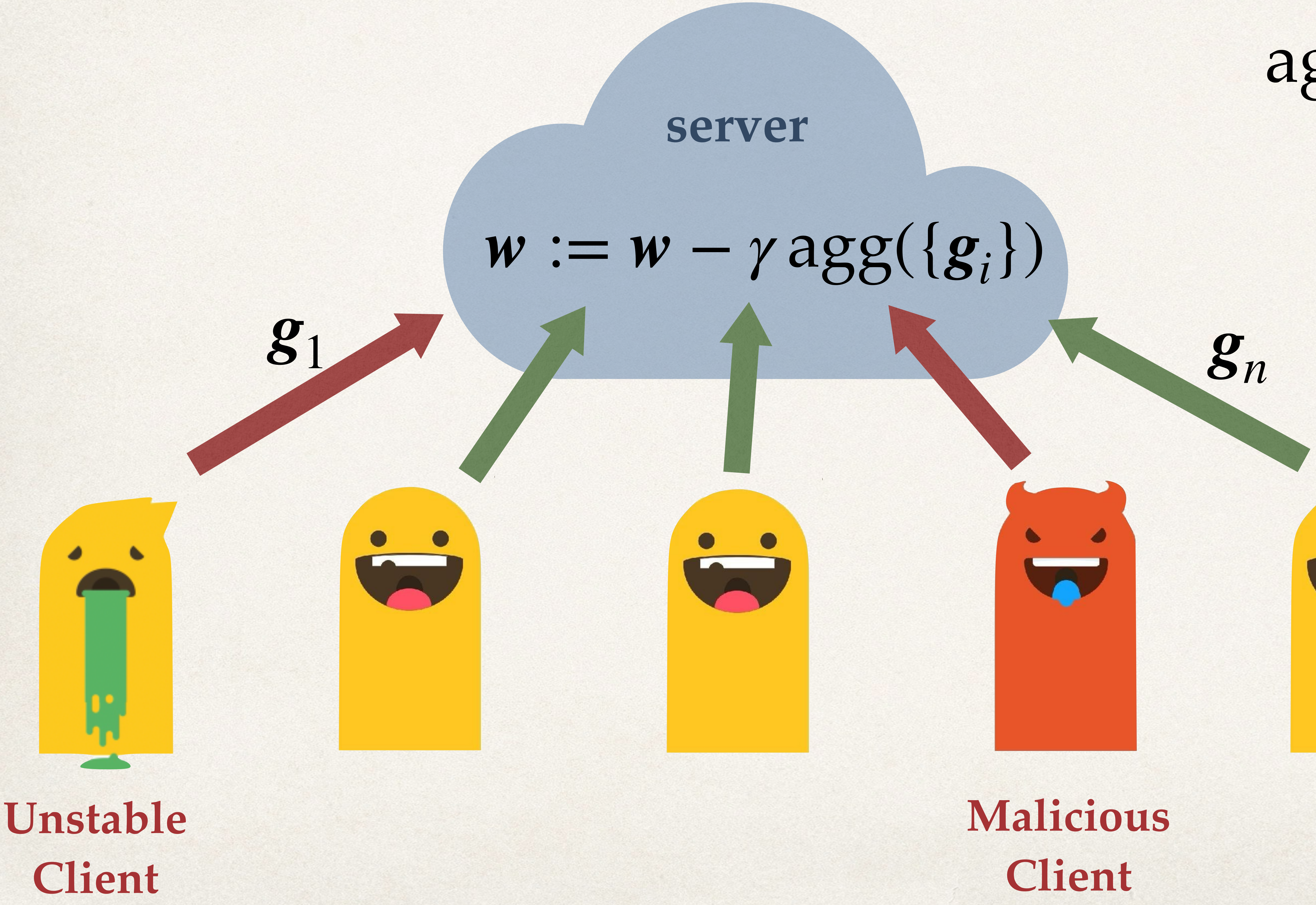
Gradients from faulty/malicious collaborators:

- Byzantine-robust Training

Malicious actors in FL



Byzantine Robust Training



$$\mathrm{agg}(\{\boldsymbol{g}_i\}) := \mathrm{avg}(\{\boldsymbol{g}_i\})$$

$$:= \mathrm{CM}(\{\boldsymbol{g}_i\})$$

Examples:

- Coordinate-wise median [Yin et al. 2017]
- Krum [Blanchard et al. 2018]
- Geometric median / RFA [Pillutla et al. 2019]

Byzantine-robust training

- **Mean vs median**

Zeno: Byzantine-suspicious stochastic gradient descent, 2018, Cong Xie et al.

Negative result

- Robustness of the aggregation rule $\mathrm{agg}(\{\boldsymbol{g}_i\})$ does **not** imply robust training: *time-coupled attacks - "little is enough"*

- Any aggregation rule which does not use history can **fail** for training (convergence)

Fix: Using history with momentum

- Simply use worker momentum

$$\boldsymbol{m}_i := (1-\beta)\boldsymbol{g}_i + \beta\boldsymbol{m}_i$$

- Effectively averages past gradients, reducing variance

- (Robustly) aggregate worker momentum instead of gradients

$$\boldsymbol{w} := \boldsymbol{w} - \gamma\,\mathrm{agg}(\{\boldsymbol{m}_i\})$$

Robustness vs Fairness

	Robust mean	**Federated**	**Fairness**
Objective	$\text{robust-mean}_i \; f_i(\boldsymbol{x})$ $= \frac{1}{\lvert good \rvert} \sum_{i \in good} f_i(\boldsymbol{x})$	$\frac{1}{n} \sum_{i}^{n} f_i(\boldsymbol{x})$	$\max_i f_i(\boldsymbol{x})$

3b Adversarial Attacks (at inference time)

Image: [Tom B. Brown/Dandelion Mané]()

Image: Elsayed ,Papernot et al 2018

Adversarial Attacks (at inference time)

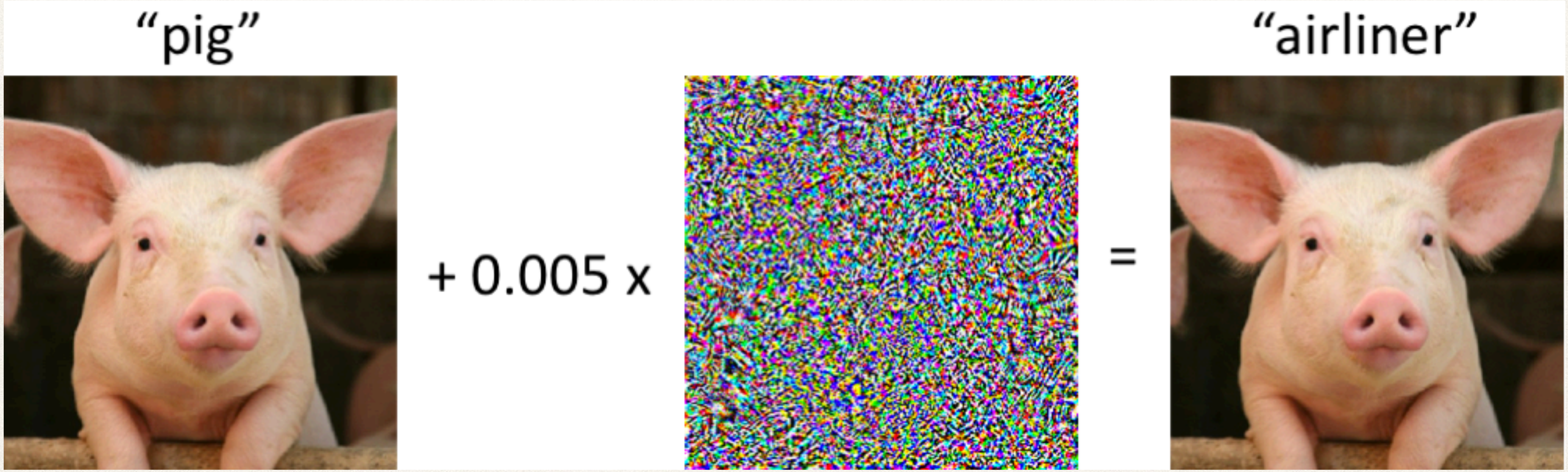


Image: [Mądry, Schmidt]

More info:
http://gradientscience.org/intro_adversarial/

Adversarial Attacks

- Standard **training**

$$\min_{\mathbf{w}} \quad f_{\mathbf{w}}(\mathbf{x}_i)$$

$\nabla_{\boldsymbol{w}} f$

change **model**

- **Attacking**

$$\max_{\mathbf{e} \in R_\infty(0,\varepsilon)} f_{\mathbf{w}}(\mathbf{x}_i + \mathbf{e})$$

$\nabla_{\boldsymbol{x}_i} f$

change **data**

- by **Projected Gradient Descent!**

4

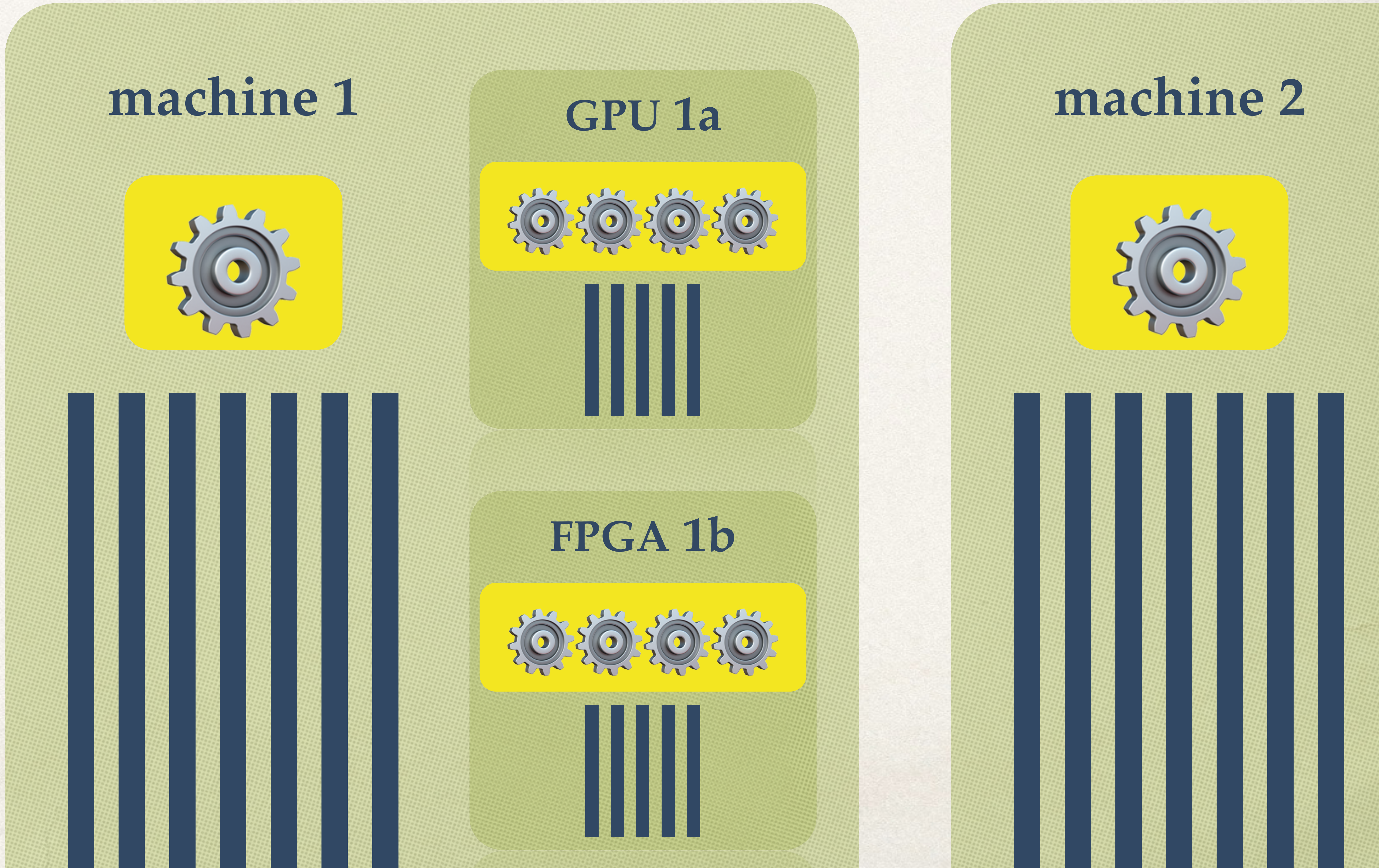
Privacy

- Secure Multiparty Computation
    - secure aggregation
      (private gradients, public model)
- Differential Privacy
- Privacy/inference Attacks

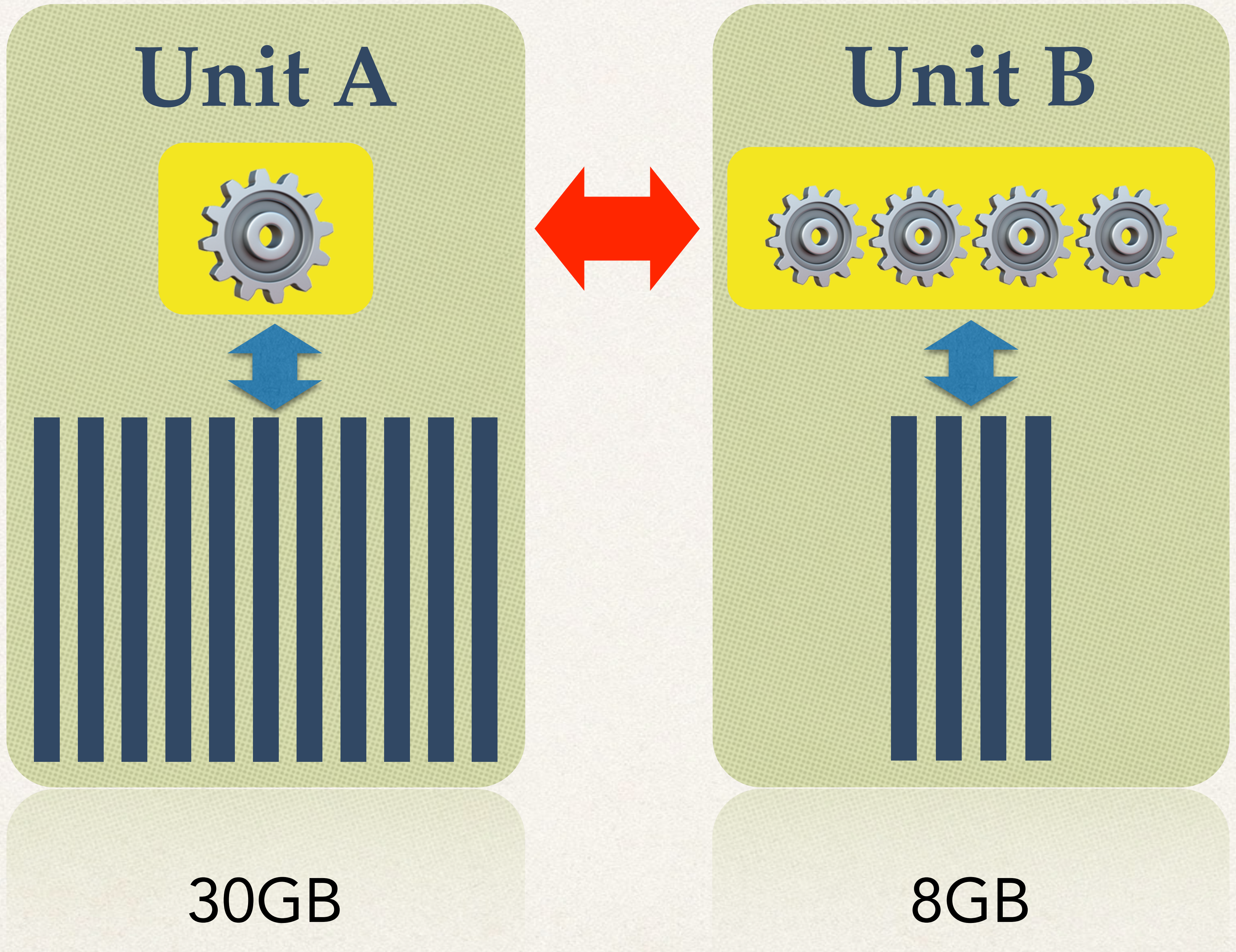
5

Leveraging Heterogenous Systems

Compute & Memory Hierarchy: Which data to put in which device?



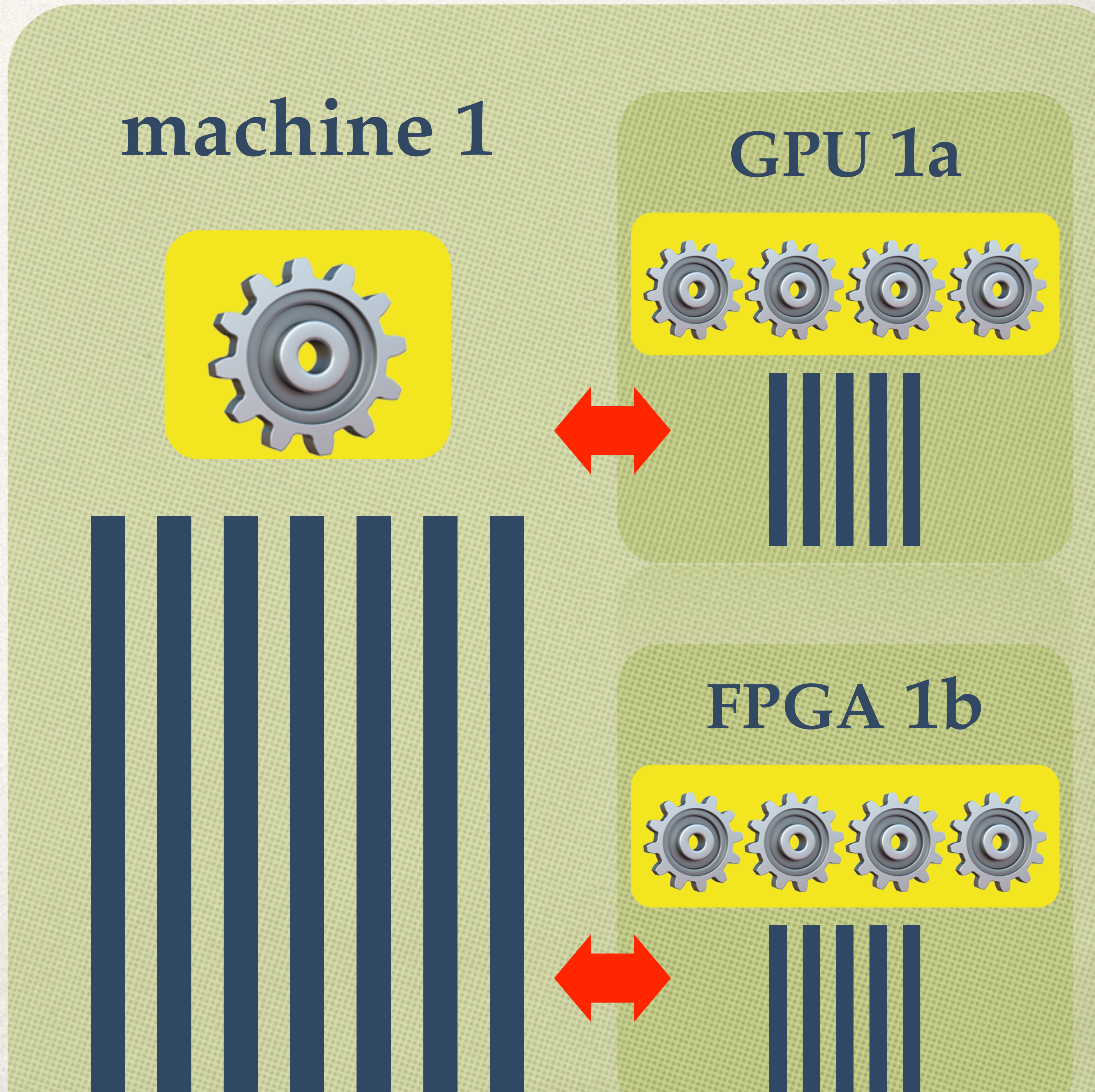
Leveraging Heterogenous Systems



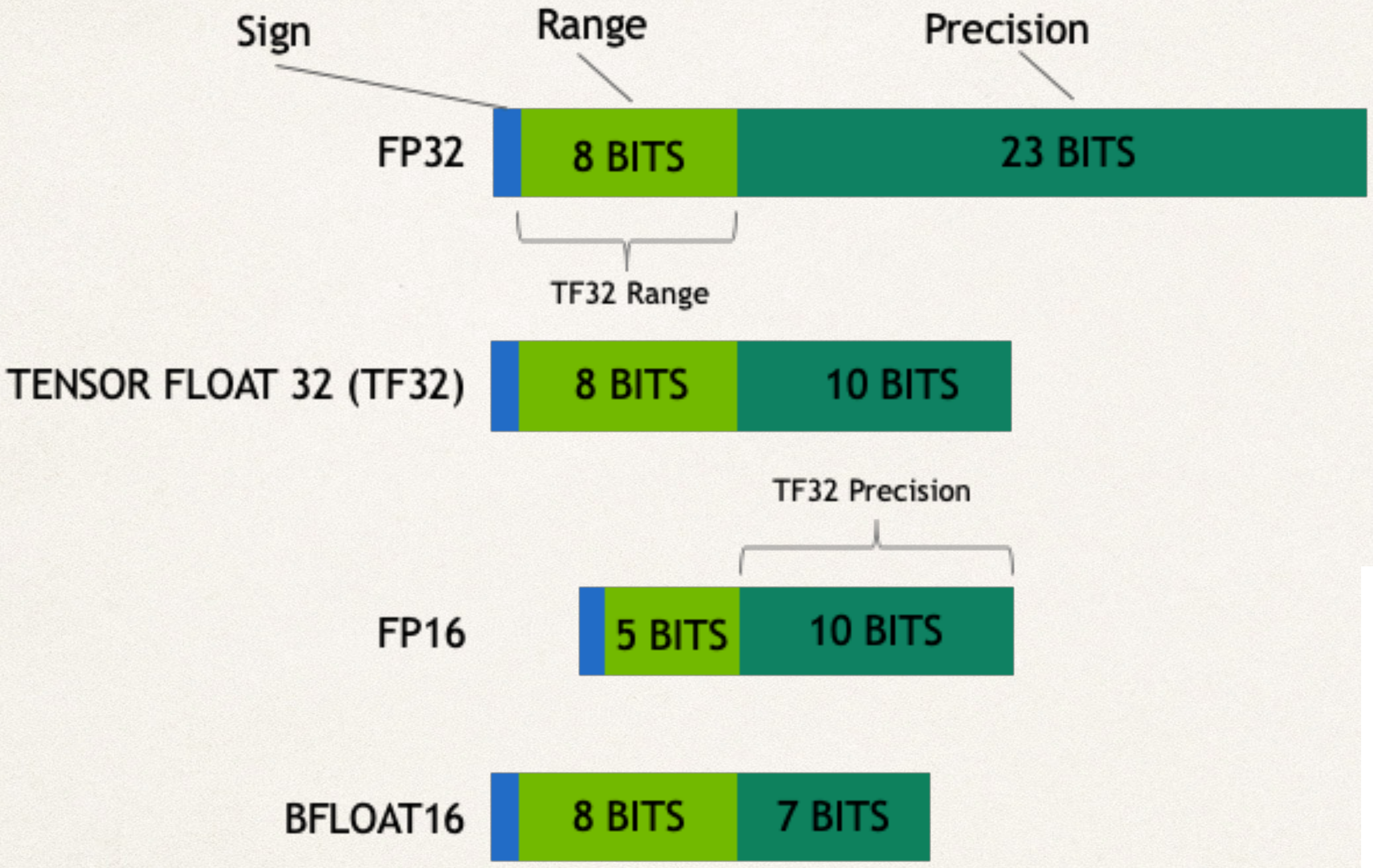
adaptive importance sampling of datapoint
e.g. for general linear models, or word2vec

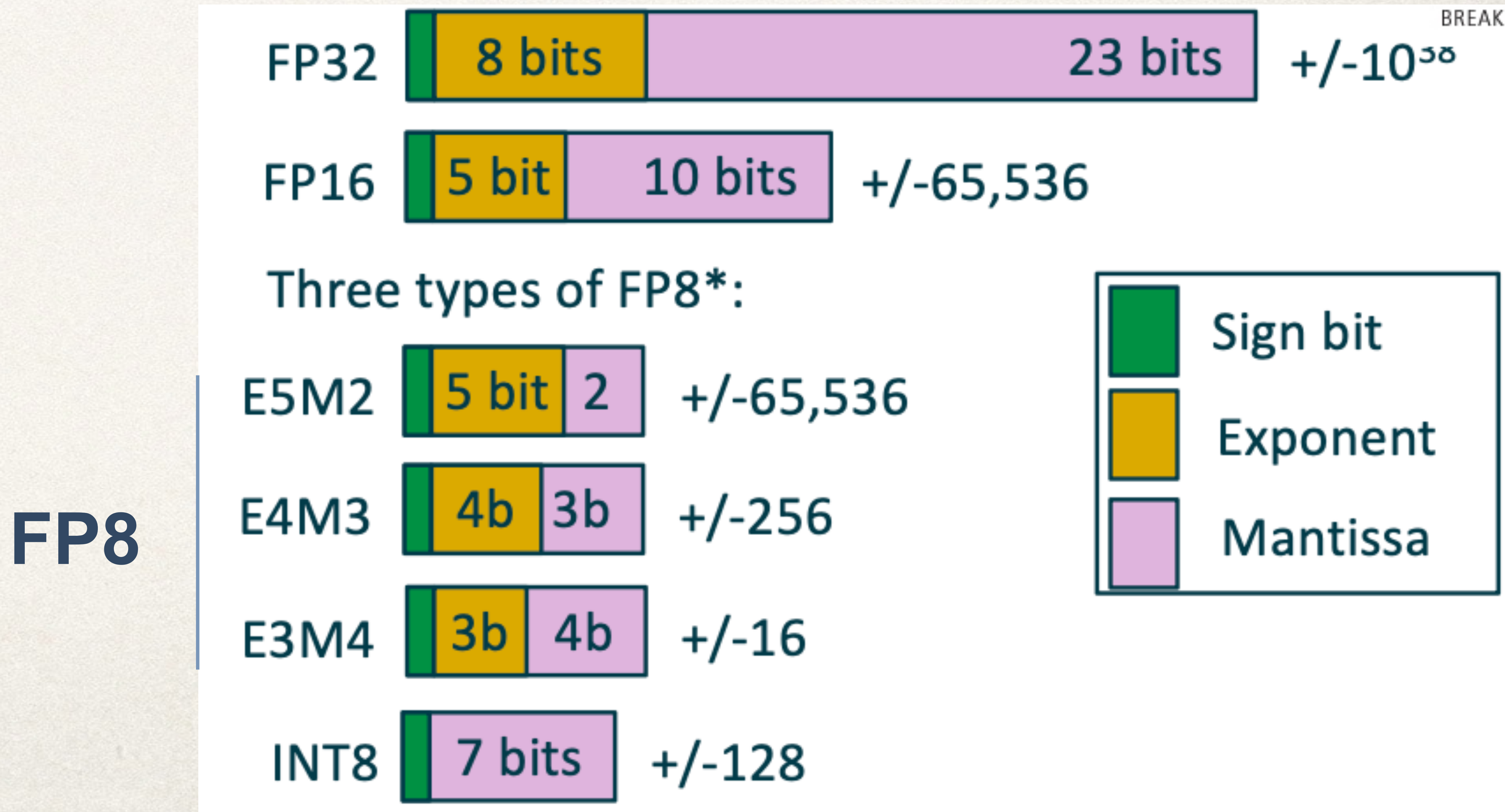
Trends - Systems

- new **hardware**
  - efficient numerics (limited precision), model compression
  - sparse ops
  - Groq, TPU, Cerebras
- **Software** frameworks
  - AutoGrad (Jax, PyTorch, TensorFlow etc)
  - Backends for new hardware



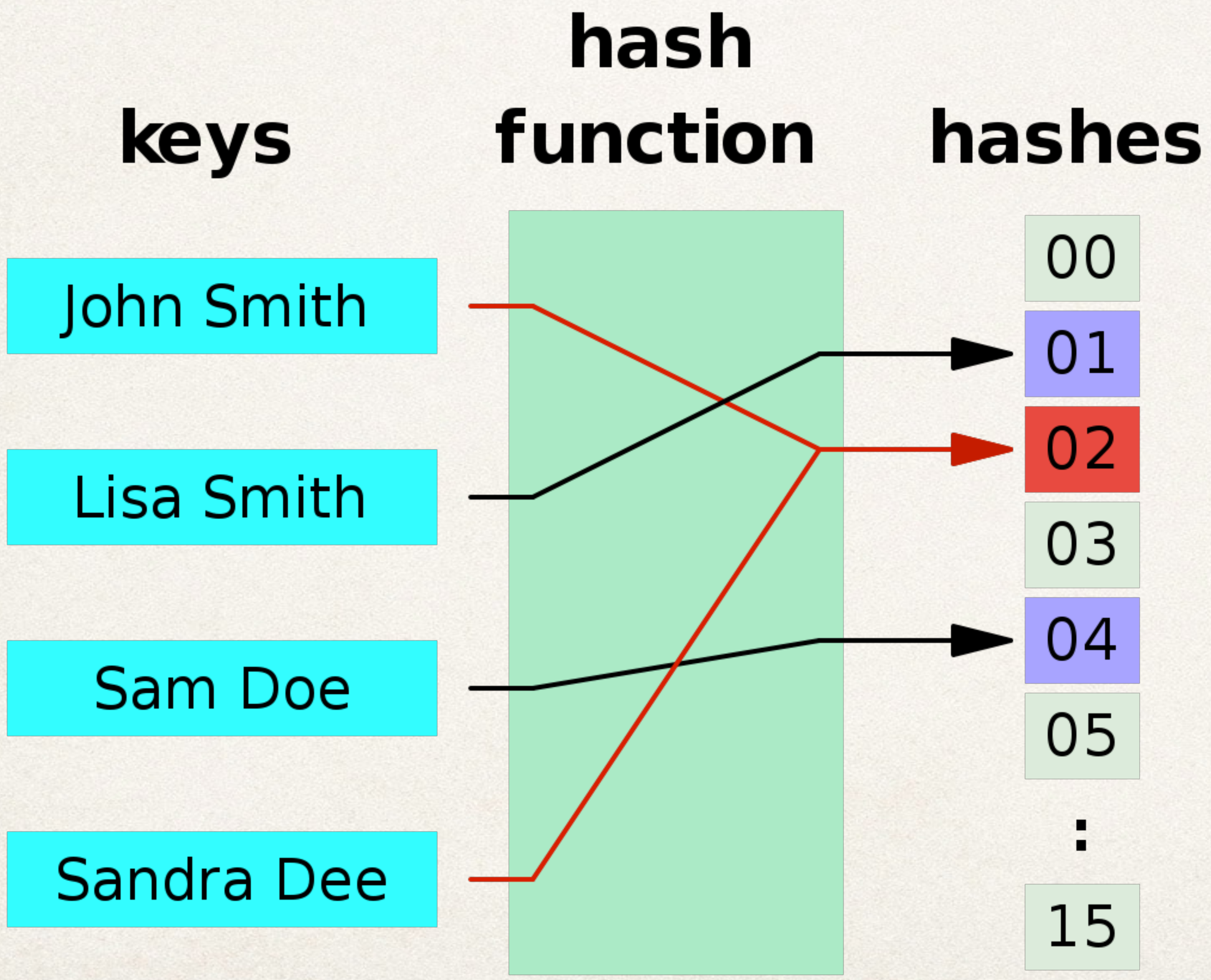
Number formats for DL





Practical tricks

- feature hashing



- limited precision operations

Meta-learning & Auto ML

- **learning to learn**
  adaptive methods
- **hyper-parameter optimization**
  zero-order methods
- **neural architecture search**
  zero-order, warm-start

Thanks!

mlo.epfl.ch

tml.epfl.ch