Proceedings of the 33rd ACM Web Conference 2024

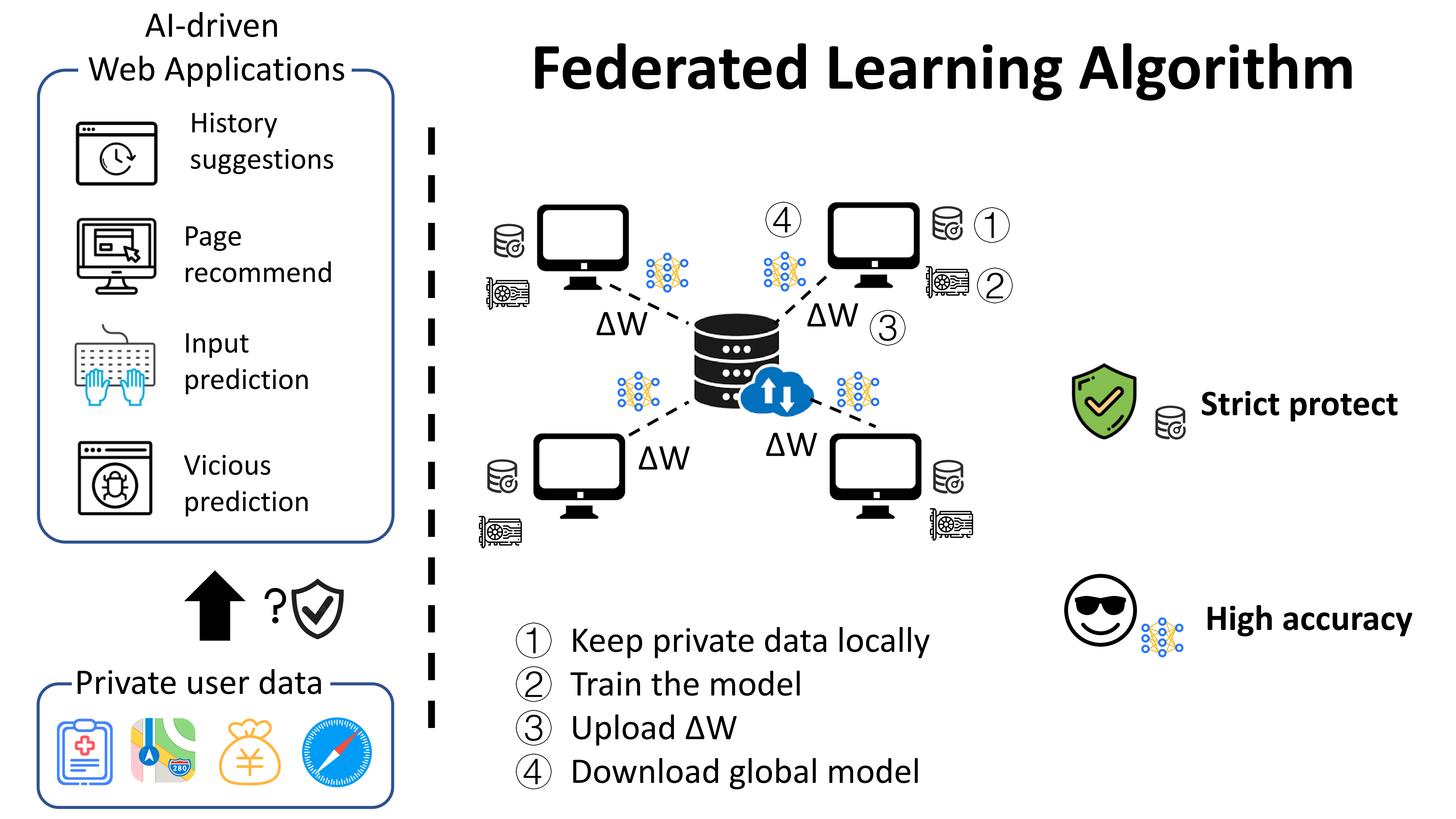
Towards Energy-efficient Federated Learning via INT8-based Training on Mobile DSPs

Jinliang Yuan, Shangguang Wang, Hongyu Li, Daliang Xu, yuanchun Li, Mengwei Xu, Xuanzhe Liu

Peking University

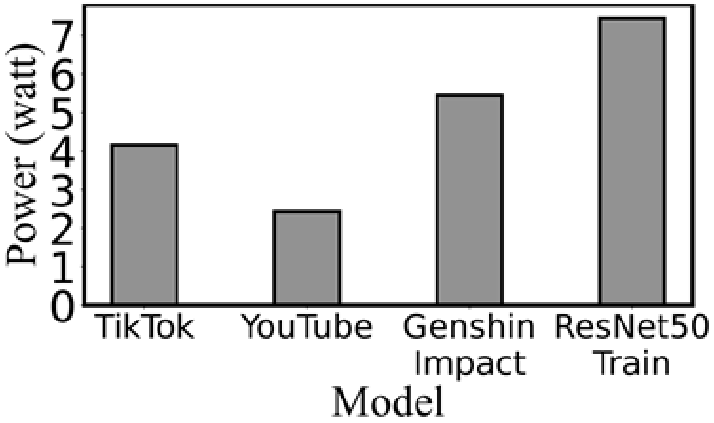
Beijing University of Posts and Telecommunications

Tsinghua University


AI-driven
Web Applications
History suggestions
Page recommend
Input prediction
Vicious prediction
?
Private user data
280
Federated Learning Algorithm
ΔW
ΔW
ΔW
ΔW
④
①
②
③
① Keep private data locally
② Train the model
③ Upload ΔW
④ Download global model
Strict protect
High accuracy

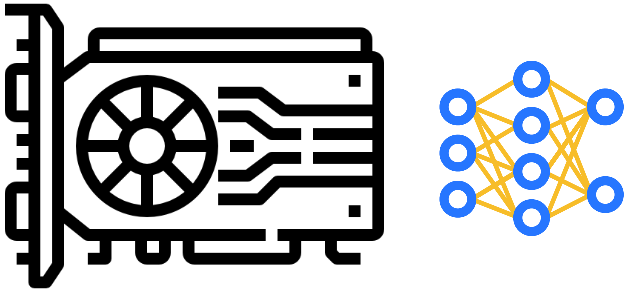
Is Federated Learning Algorithm enough to protect data?

No! The **energy** is the main fence.

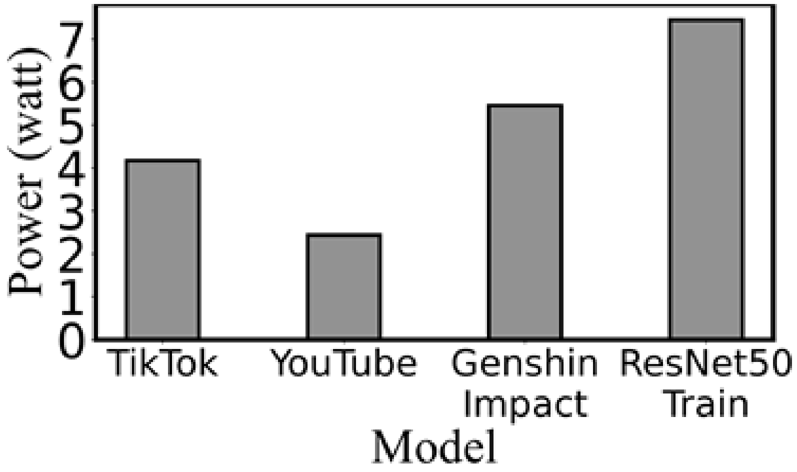


Such huge **energy** is unacceptable.

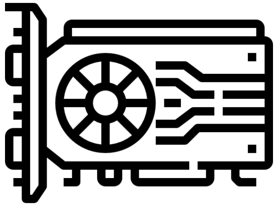
E(Training VGG16 on CIFAR-10) = **E**(Watch TikTok/YouTube for 12/24 hours) = **E**(Play video game for 9 hours)



Why energy consumption is so huge?

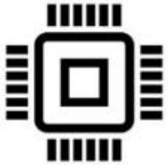


*DNN models are trained on **GPUs***



- floating computation
- matrix manipulation

*For phones, they need executing on **CPUs***



- integer computation
- general computing

We observe there is **DSP** on the phone

It’s energy friendly and can handle matrix manipulation

Models	CPU, FP32		CPU, INT8		DSP, INT8	
	T	E	T	E	T	E
MobileNet-V1	11.4	88.2	4.3	22.7	2.5	5.4
MobileNet-V2	8.6	64.8	4.6	22.9	3.1	5.4
ResNet-50	78.7	597.6	27.8	131.4	9.2	28.8
Inception-V4	266.3	1,980	81.5	399.6	17.2	59.4
EfficientNet-V2	33.0	187.2	13.4	59.4	8.44	12.6

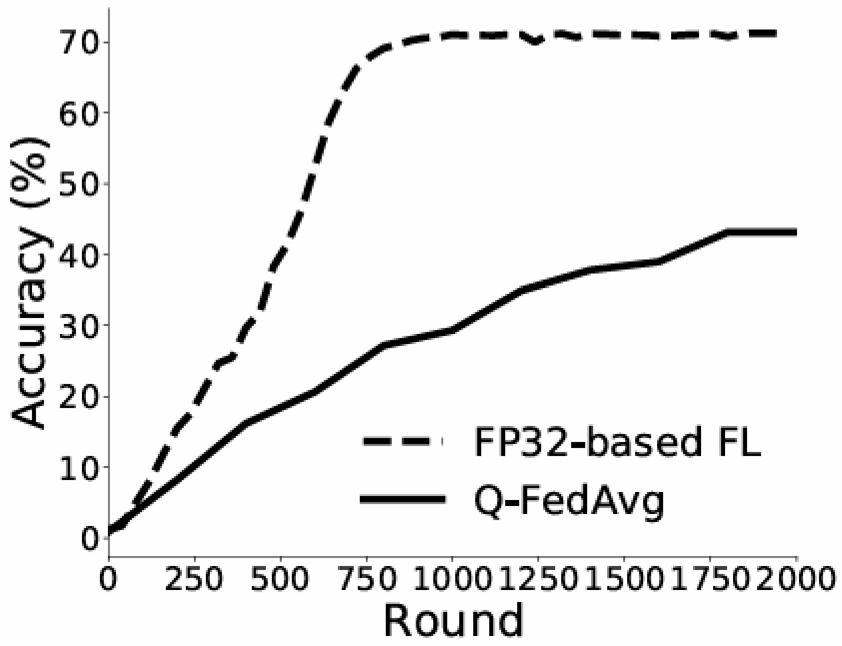
	FP32 Acc. (%)	Octo Acc. (%)	Acc. Degradation (%)
GoogLeNet, FM	99.1 – 99.5	97.9 – 98.6	0.9 – 1.2
GoogLeNet, CF	97.8 – 99.2	97.6 – 98.8	0.2 – 0.4
AlexNet, FM	95.6 – 98.4	92.8 – 94.3	2.8 – 4.1
AlexNet, CF	91.8 – 95.2	86.1 – 87.3	5.7 – 7.9
VGG11, FM	97.5 – 98.8	94.4 – 96.5	2.3 – 3.1
VGG11, CF	97.2 – 99.5	96.5 – 98.6	0.7 – 0.9

Save energy 32X, speeds up 9X

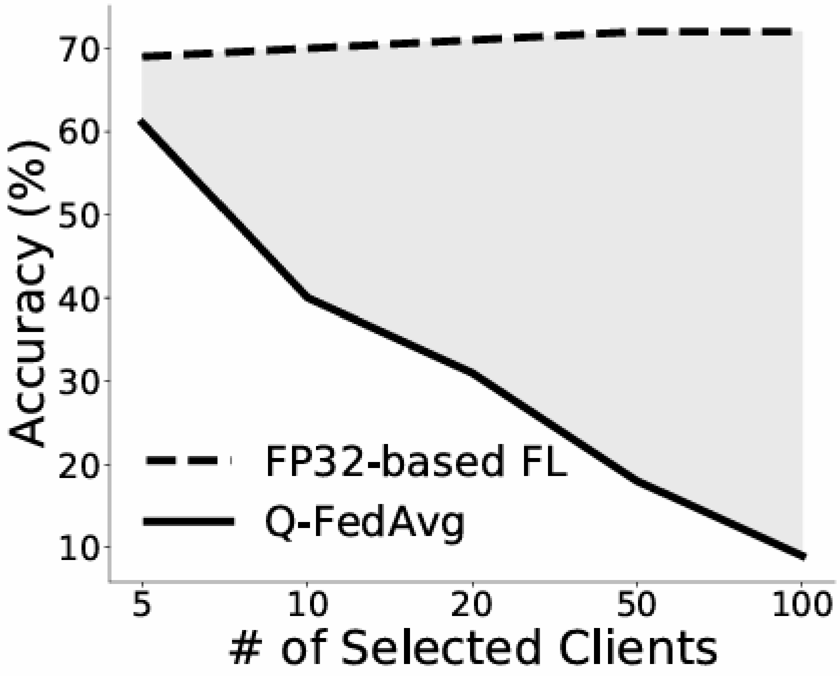
Accuracy losses only 2%

Challenge: Directly using DSP with FL is impractical

Low and slow convergence

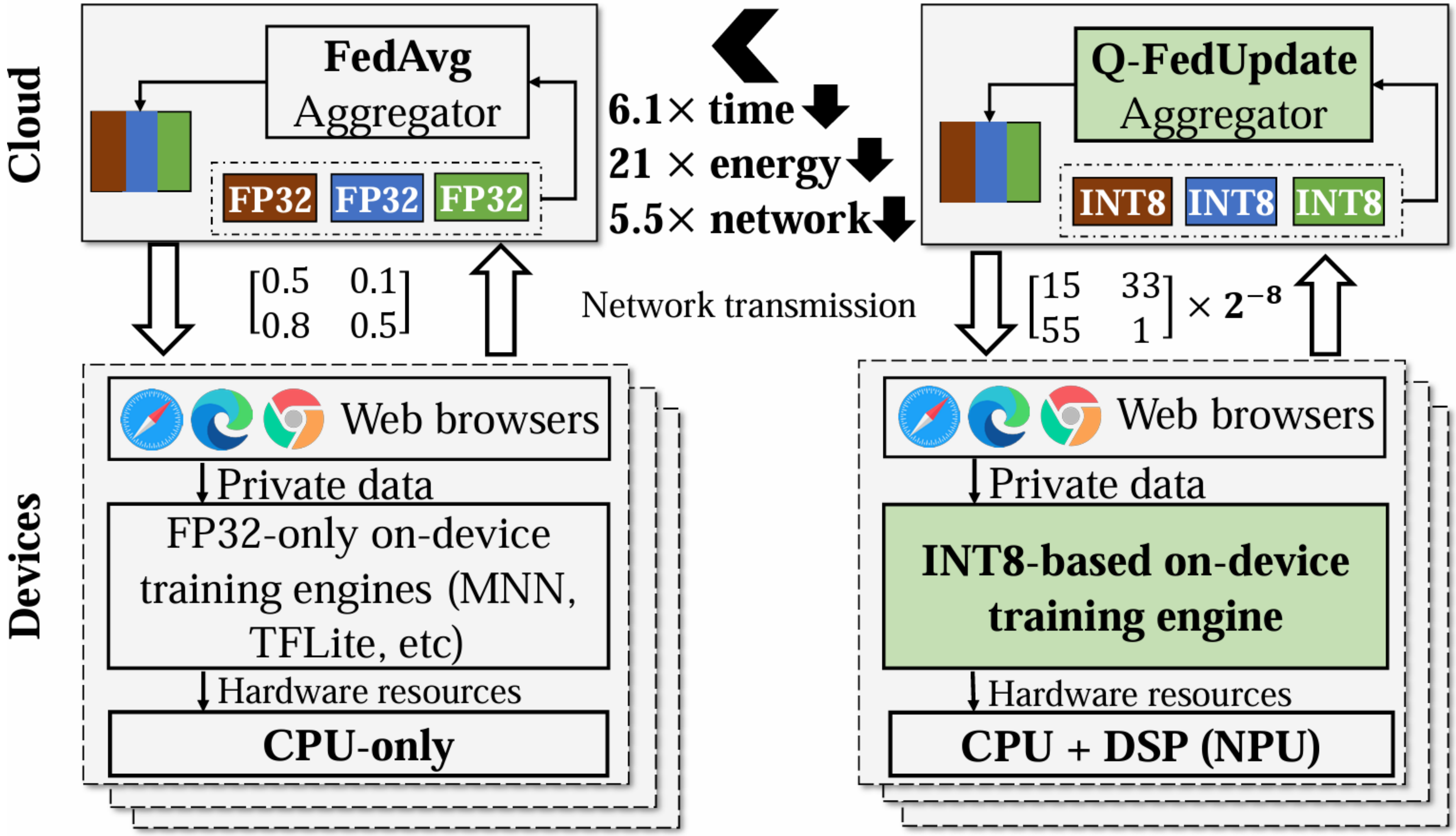


Low scalability



GOAL: How to design an **energy efficient** and **high accuracy** algorithm with DSP?

Key Idea: FP32-INT8



(a) Traditional FP32-based FL (b) Proposed INT8-based FL

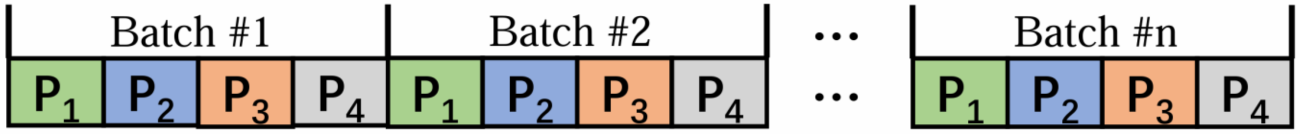
Design1:

Error-Compensated Aggregation

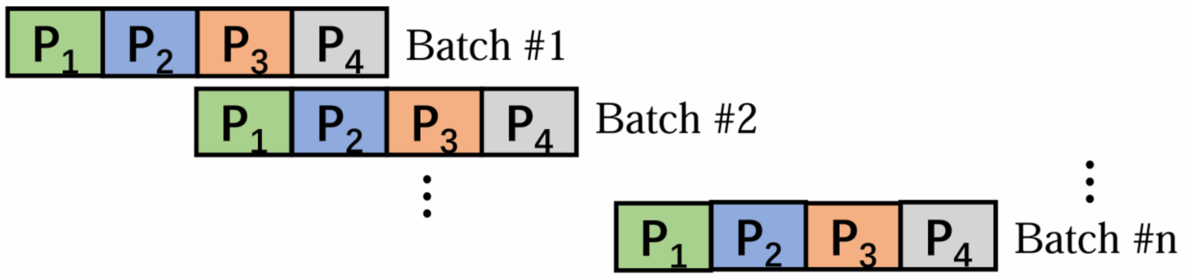
$$
\begin{aligned}
w(t+1) &= w(t) - \sum_{k=1}^{K} \frac{n_k}{n} \Delta w_d^k(t+1) \\
&= w(t) - \sum_{k=1}^{K} \frac{n_k}{n} \left(w_d(t) - w_d^k(t+1)\right) \\
&= \underbrace{\boxed{\underline{w(t) - w_d(t)}}}_{\text{Quantization Error}} + \underbrace{\boxed{\sum_{k=1}^{K} \frac{n_k}{n} w_d^k(t+1)}}_{\text{Q-FedAvg}}.
\end{aligned}
$$

Design2:

Pipelined Batch Quantization



(a) Traditional four procedures of local training on device



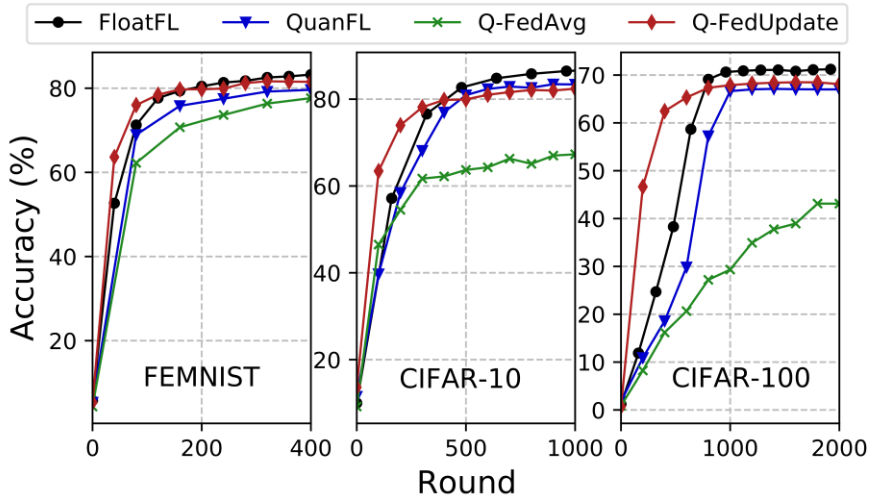
(b) Pipeline procedures: parallelization of P_2 on CPU and P_4 on DSP

Evaluation Results

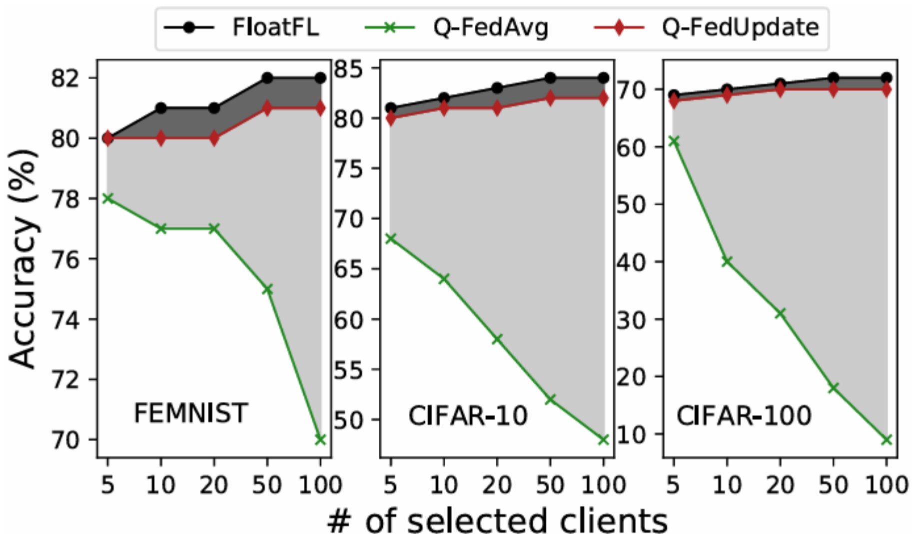
Algorithms	FEMNIST			CIFAR-10			CIFAR-100		
	Acc (%)	T (hours)	E (kJ)	Acc (%)	T (hours)	E (kJ)	Acc (%)	T (hours)	E (kJ)
FloatFL (#1)	**82 (1↓)**	0.57	0.56	**84 (3↓)**	8.94	42.5	**71 (2↓)**	7.58	2.2
QuanFL (#1)	80	0.40	0.62	82	6.17	53.1	67	3.09	2.6
Q-FedUpdate (#2)	81	0.14	0.18	81	2.80	18.9	69	1.71	0.9
Q-FedUpdate (L)	81	**0.12 (4.8×)**	**0.04 (14×)**	81	**2.40 (3.7×)**	**4.6 (9×)**	69	**1.61 (4.7×)**	**0.21 (10×)**
Q-FedUpdate (M)	81	**0.08 (7.1×)**	**0.02 (28×)**	81	**1.77 (5.1×)**	**3.0 (14×)**	69	**1.47 (5.2×)**	**0.14 (16×)**
Q-FedUpdate (H)	81	**0.07 (8.1×)**	**0.02 (28×)**	81	**1.62 (5.5×)**	**2.7 (16×)**	69	**1.44 (5.3×)**	**0.13 (17×)**

Time speeds up 6x

Energy saves 21X



Accuracy losses only 2%



Thanks

Q&A