

# Communication-Efficient Learning of Deep Networks from Decentralized Data

Federated Learning: invented by Google, circa 2017

<https://arxiv.org/abs/1602.05629>

<https://ai.googleblog.com/2017/04/federated-learning-collaborative.html>

---

## Problem

- Machine learning on mobile devices
- Processing large amounts of data
  - Non-IID: a particular user's data set will not be representative of population
  - Unbalanced: the amounts of local training data varies
  - Massively Distributed: number of clients participating is much larger than the average number of examples per client
- Limited communication from devices
- Handling privacy sensitive data

## Solution

Instead of using a centralized datacenter and learning after aggregating, do the opposite.

- Leave the training data distributed on the devices
- Limit attack surface to only the device not it and the cloud
- Aggregate locally computed updates on a shared model

“since the learning task is solved by a loose federation of participating devices (which we refer to as clients) which are coordinated by a central server. Each client has a local training dataset which is never uploaded to the server. Instead, each client computes an update to the current global model maintained by the server, and only this update is communicated”

This decentralized approach is referred to as “federated learning” or “FL”.

## Practicality Issues of Solution

- Computing costs
- Communication costs

When using a centralized data center, Computation costs are high while communication costs are small. However, when using a decentralized system, computation costs are essentially free while communication is now costly.

## Pseudocode

Federated Machine Learning = Stochastic Gradient Descent + Model Averaging

Synchronous update scheme:

- Server sends the current global state to each of a sample of clients (e.g., the current model parameters)
- Each selected client then performs local computation based on the global state and its local dataset, and sends an update to the server.
- Server then applies these updates to its global state, and the process repeats.

Baseline algorithm: FederatedSGD (or FedSGD), full-batch (non-stochastic) gradient descent

Interested algorithm: FederatedAveraging (or FedAvg), testing different client sample sizes

---

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

---

**Server executes:**

initialize  $w_0$

**for** each round  $t = 1, 2, \dots$  **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$  (random set of  $m$  clients)

**for** each client  $k \in S_t$  **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$w_{t+1} \leftarrow \sum_{k=1}^K \frac{n_k}{n} w_{t+1}^k$

**ClientUpdate**( $k, w$ ): *// Run on client  $k$*

$\mathcal{B} \leftarrow$  (split  $\mathcal{P}_k$  into batches of size  $B$ )

**for** each local epoch  $i$  from 1 to  $E$  **do**

**for** batch  $b \in \mathcal{B}$  **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

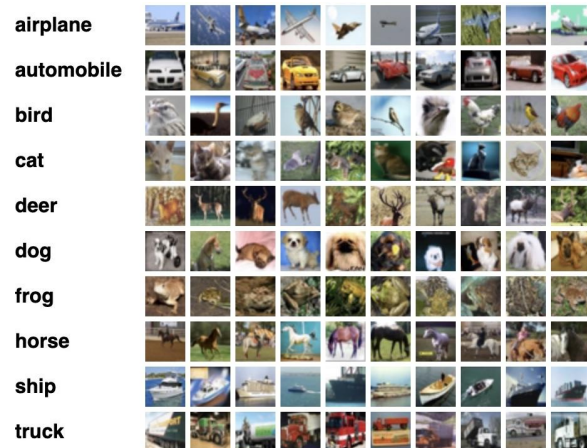
return  $w$  to server

---

## Datasets Used:

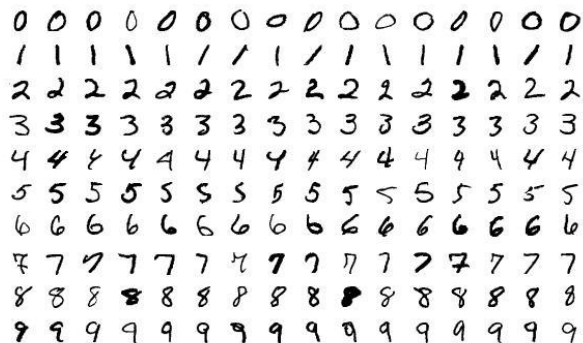
CIFAR-10 dataset (Canadian Institute for Advanced Research, 10 classes)

<https://paperswithcode.com/dataset/cifar-10>



MNIST database (Modified National Institute of Standards and Technology database)

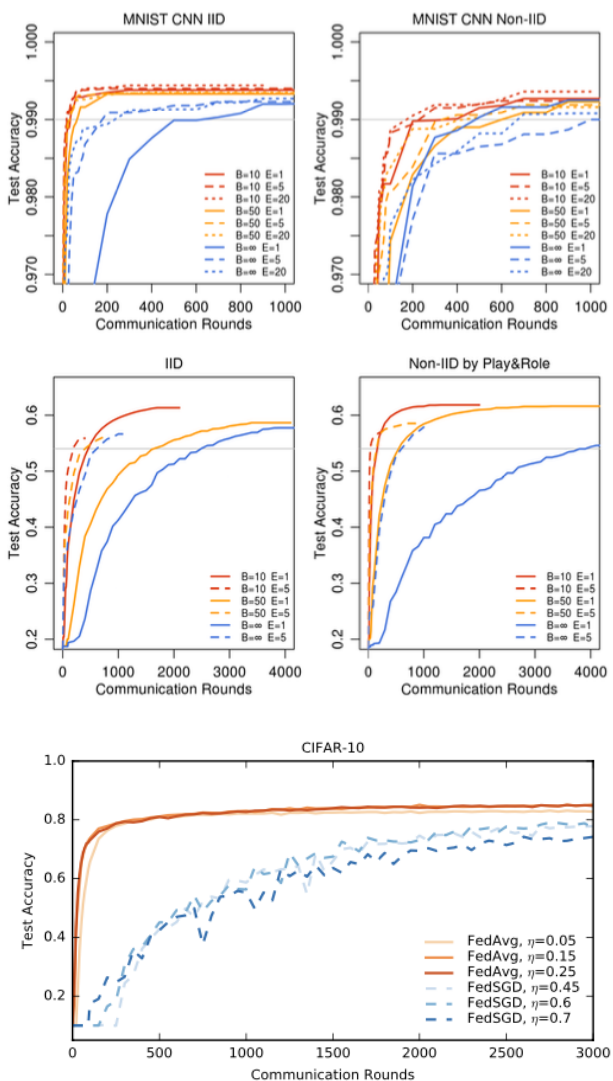
<https://paperswithcode.com/dataset/mnist>



“For language modeling, we built a dataset from *The Complete Works of William Shakespeare*”:

## Conclusions:

I guess it works.



“Our experiments show that federated learning can be made practical, as FedAvg trains high-quality models using relatively few rounds of communication, as demonstrated by results on a variety of model architectures: a multi-layer perceptron, two different convolutional NNs, a two-layer character LSTM, and a large-scale word-level LSTM.”

## Future Work:

providing stronger privacy guarantees:

- differential privacy
- secure multi-party computation, etc.