Federated Learning Paper Sharing

Lisen Dai

November 27, 2020

FedOpt: Towards Communication Efficiency and Privacy Preservation in Federated Learning

Sparse Compression Algorithm

Goal: reduce the number of communication bits during the models training.

$$\Delta\theta = \mathcal{SGD}_n(\theta, D_{mini-batches}) - \theta$$

θ: Deep Neural Network parameters.
$\mathcal{SGD}_n$: refers to the set of gradient updates after n epochs of SGD on DNN (deep neural network) parameters θ during the sampling of mini-batches from local data
Once we have the updates $\Delta\theta$...

FedOpt: Towards Communication Efficiency and Privacy Preservation in Federated Learning

Input: temporal vector $\Delta\theta$, Sparsity Fraction q
Output: sparse temporal $\Delta\theta^*$

Initialization;
$num^+ \leftarrow top_q(\Delta\theta)$; $num^- \leftarrow top_q(-\Delta\theta)$
$\Psi^+ \leftarrow mean(num^+)$; $\Psi^- \leftarrow mean(num^-)$
if $\Psi^+ \geq \Psi^-$ **then**
  return $(\Delta\theta^* \leftarrow \Psi^+(\theta \geq \min(num^+)))$;
end
else
  return $(-\Delta\theta^* \leftarrow \Psi^-(\theta \geq \min(-num^-)))$;
end

Algorithm 1: SCA: Communication Efficiency in FedOpt

FedOpt: Towards Communication Efficiency and Privacy Preservation in Federated Learning

"We utilise the additively homomorphic encryption in FedOpt in order to achieve efficiency throughout the learning process."

Algorithm 2: Pseudocode of Privacy Preserving

Input : Users for local datasets D_i, the cloud server to initialise global parameters ϖ_o
Output: New global parameters ϖ

Initialisation:
while *Cloud server initialise global parameters* ϖ_o **do**
  Aggregate global parameters ϖ_o to users
  while *Users obtain local gradients* G_{II} *by training local models* D_i **do**
    Add noise ϵ-DP $\leftarrow G_{\text{II}}$
    Encrypt $G_{\text{II}} \leftarrow E_\delta(G_{\text{II}} + Lap(\frac{\Delta f_{\text{II}}}{\epsilon}))$
    Generate encrypted local gradients E_{II}
    Aggregate $E_\delta(\sum_{\text{II}=1}^{n} G_{\text{II}})$
  end
  while *Cloud server aggregates encrypted local gradients to users* II **do**
    $E_{add} \leftarrow E_\delta(\sum_{\text{II}=1}^{n} G_{\text{II}})$
    Generate cipher-text from E_{II}
    Generate encrypted global gradients E_{add}
  end
  while *Users decrypts* E_{add} *to get global gradients* B_{II} **do**
    $D_\delta(E_{add}) \leftarrow \sum_{\text{II}=1}^{n} G_{\text{II}}$
    Update existing parameters ϖ
    Aggregate new parameters ϖ to the cloud server

FedOpt: Towards Communication Efficiency and Privacy Preservation in Federated Learning

Algorithm 3: FedOpt: Communication-Efficiency and Privacy-Preserving

Input : Initial parameters ϖ

Output: Global model with improved parameters ϖ_o

Initialisation: all users $\amalg_i, i = 1, \ldots,$ [Total number of users] are initialised with the same parameters $\nu_i \leftarrow \nu$. Those users who carry different private datasets D_i with $|\{c : (x, y) \in D_i\}|$ = [total classes per user]. The remaining $\amalg$ are initialised to zero $\Delta\nu, \mathcal{R}_i, \mathcal{R} \leftarrow 0$.

for *epoch* $e = 1, \ldots, E$ | E = *Total number of Epochs* | **do**

  for $\amalg_i \in \amalg \subseteq \{1, \ldots,$ *[Number of users]*$\}$ **do**

    User $\amalg_i$ execute:

    Plain-text = $\xi \leftarrow downloads_{CS \rightarrow \amalg_i}(\xi)$

    $\Delta\nu \leftarrow$ decrypt (ξ)

    $\nu_i \leftarrow \nu_i + \Delta\nu$

    $\Delta\nu_i \leftarrow \mathcal{R}_i + SGD(\nu_i, D_i) - \nu_i$

    $\Delta\overline{\nu_i} \leftarrow SCA_{upload}(\Delta\nu_i)$

    $\mathcal{R}_i \leftarrow \Delta\nu_i - \Delta\overline{\nu_i}$

    $\xi_i \leftarrow$ encrypt $\Delta\overline{\nu_i}$

    $upload_{\amalg_i \rightarrow CS}(\xi_i)$

  end

  Cloud Server CS execute:

  $collect_{\amalg_i \rightarrow CS}(\Delta\overline{\nu_i}), e \in \amalg$

  $\Delta\nu \leftarrow \mathcal{R} + \frac{1}{\amalg} \sum_{e \in \amalg} \Delta\overline{\nu_i}$

  $\Delta\overline{\nu} \leftarrow SCA_{download}(\Delta\nu)$

  $\mathcal{R} \leftarrow \Delta\nu - \Delta\overline{\nu}$

  $\nu \leftarrow \nu + \Delta\overline{\nu_i}$

FedOpt: Towards Communication Efficiency and Privacy Preservation in Federated Learning

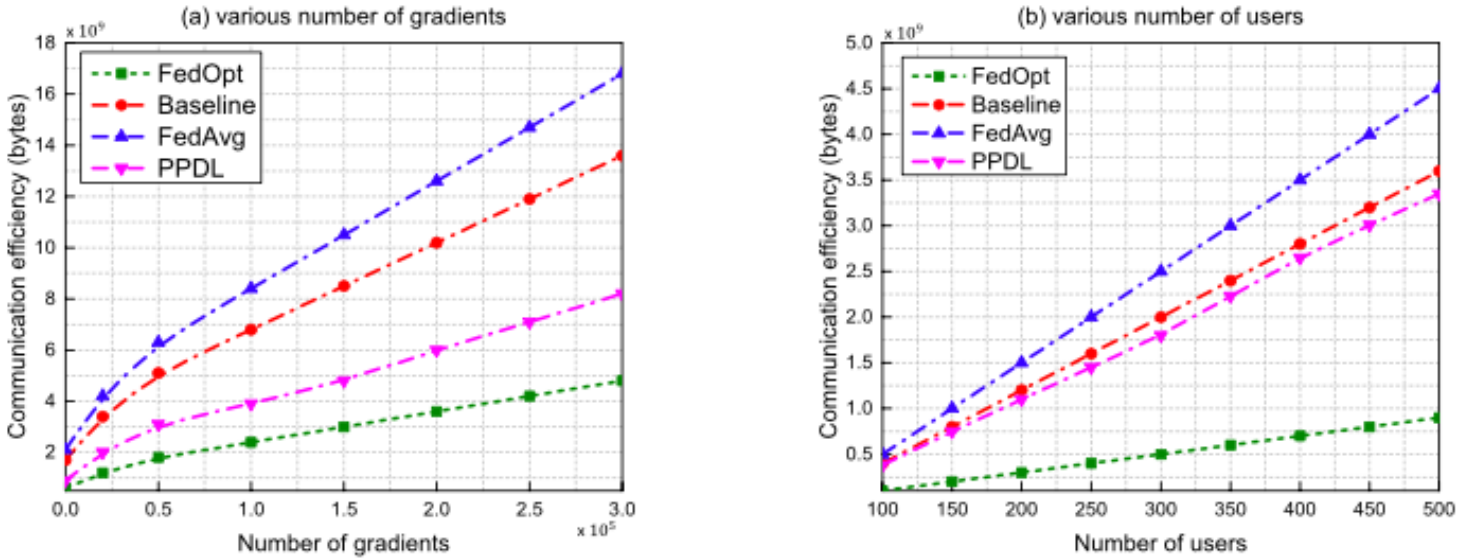
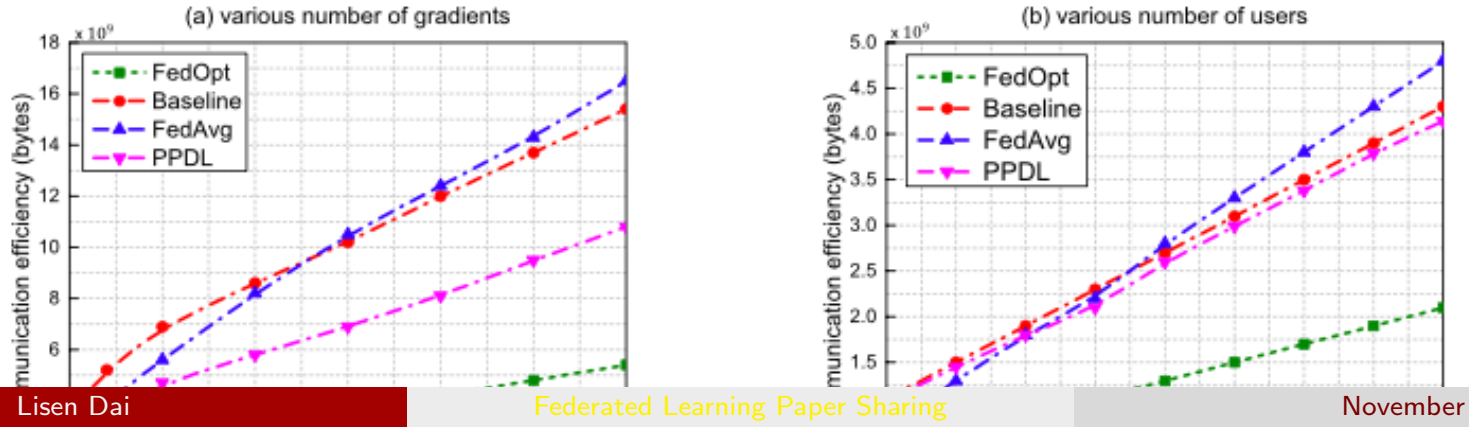


Figure 5. FedOpt communication efficiency on MNIST dataset.



FedOpt: Towards Communication Efficiency and Privacy Preservation in Federated Learning

Table 2. Communication bits required for upload and download to achieve the targeted accuracy.

	MNIST (Accuracy = 91.3)	**CIFAR-10 (Accuracy = 87.6)**
Baseline	2218/2218 MB	35653 MB/35653 MB
FedAvg *epochs* = 50	119.65 MB/119.65 MB	2589.5 MB/2589.5 MB
FedAvg *epochs* = 100	84.73 MB/84.73 MB	1665.7 MB/1665.7 MB
PPDL *epochs* = 50	98.63 MB/311.6 MB	1472.2 MB/4739.2 MB
PPDL *epochs* = 100	63.74 MB/432.2 MB	958.3 MB/6342.4 MB
FedOpt *epochs* = 50	10.2 MB/102 MB	109.23 MB/1090.3 MB
FedOpt *epochs* = 100	14.6 MB/146 MB	172.3 MB/1723 MB

One-Shot Federated Learning

Background

"In this work, we instead focus on techniques for `one-shot federated learning`, in which we learn a global model from data in the network using only a single round of communication between the devices and the central server. "
"Our key insight is that if each local device trains a local model to completion (as opposed to computing incremental updates as in traditional federated learning methods), we can effectively apply ensemble methods to capture global information across the device-specific models."

One-Shot Federated Learning

Methods

Local: for m devices, each device $t \in [m]$ possesses local data $X_t \in \mathcal{R}^{d \times n_t}$ and solves:

$$\min_{w_t \in \mathcal{R}^d} \{\mathcal{P} w_t := \frac{1}{n_t} \sum_{i=1}^{n_t} \ell_i(x_i^T + w_t) + \frac{\lambda}{2} \|w_t\|^2\}$$

where the vector $\{x_i\}_{i=1}^{n_t} \in X_t$ and ℓ_i are real-valued convex loss functions (i.e. hinge loss). In the kernelized setting, we solve the dual formulation of this problem:

$$\max_{\alpha_t \in \mathcal{R}^{n_t}} \{-\frac{1}{2\lambda n_t^2} \alpha_t^T \phi(X_t)^T \phi(X_t) \alpha_t + \frac{1}{n_t} \sum_{i=1}^{n_t} -\ell^*(-[\alpha_t]_i)\}$$

where $\phi(X_t)^T \phi(X_t)$ can be replaced with $k(X_t, X_t)$ using kernel trick.

One-Shot Federated Learning

Methods

Ensemble:

1. Cross-Validation (CV) Selection: Devices only share their local models if they achieve some baseline performance (e.g., in terms of ROC AUC) on their local validation data, with the baselines determined in advance by the server. The server ensembles the k best performing models from this subset of local models.
2. Data Selection: Devices only share their local models if they have some baseline amount of local training data, with the baseline determined in advance by the server. The server ensembles models from these local models trained on the top k largest data sets.
3. Random Selection: The server randomly selects k devices from the network and creates an ensemble from the corresponding local models.

The final ensemble F_k of k device models is constructed by averaging the predictions of each model.

One-Shot Federated Learning

Methods

Distillation (Semi-Supervised):
When the amount of clients is large...
When the central server has access to unlabelled public proxy data, F_k can be compressed into a smaller model, f', via distillation.
`New Method`: For proxy data $x'_1, ..., x'_l$, we generate corresponding "soft" labels $F_k(x')$, ...,. In particular, we perform distillation in the dual by minimizing the L2 difference in predictions between the student and teacher on the proxy data:

$$\min_{\alpha' \in \mathcal{R}^l} \frac{1}{l} \sum_{i=1}^{l} (F(x'_i) - \sum_{j=1}^{l} \alpha'_j k(x'_j, x'_i))^2$$

to produce a distilled model $f'(x) = \sum_i^l = 1 \alpha_i \phi(x'_i)$. When there are privacy concerns with sharing local models between devices (e.g., for dual SVMs, which require local support vectors to be shared), distillation not only helps to compress the model but also enables privacy-preservin learning.

One-Shot Federated Learning

Result

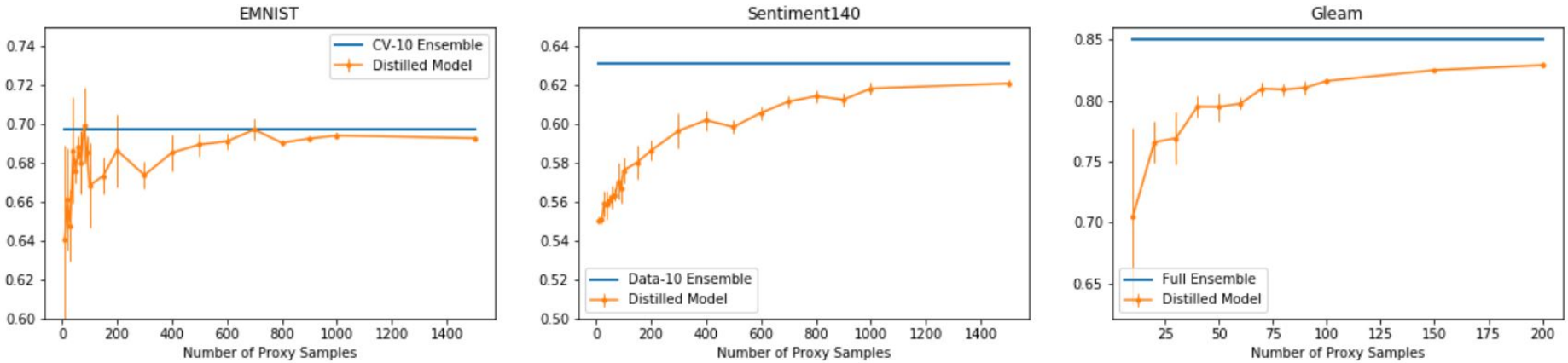


Figure 3: Comparison of distilled model and ensemble for different proxy data sizes (averaged over 5 trials). The distilled model matches the original ensemble performance with relatively few proxy samples.

MNIST Handwritten characters authored by different individuals (devices). We predict between lowercase and uppercase letters

ent140 Binary sentiment detection on tweets from different users (devices). We use a bag-of-words representation to featurize tweets

Gleam Two hours of high resolution Google Glass sensor data corresponding to different activities. We predict between eating and other activities (walking, talking, etc)

Expanding the reach of federated learning by reducing client resource requirement

Overview

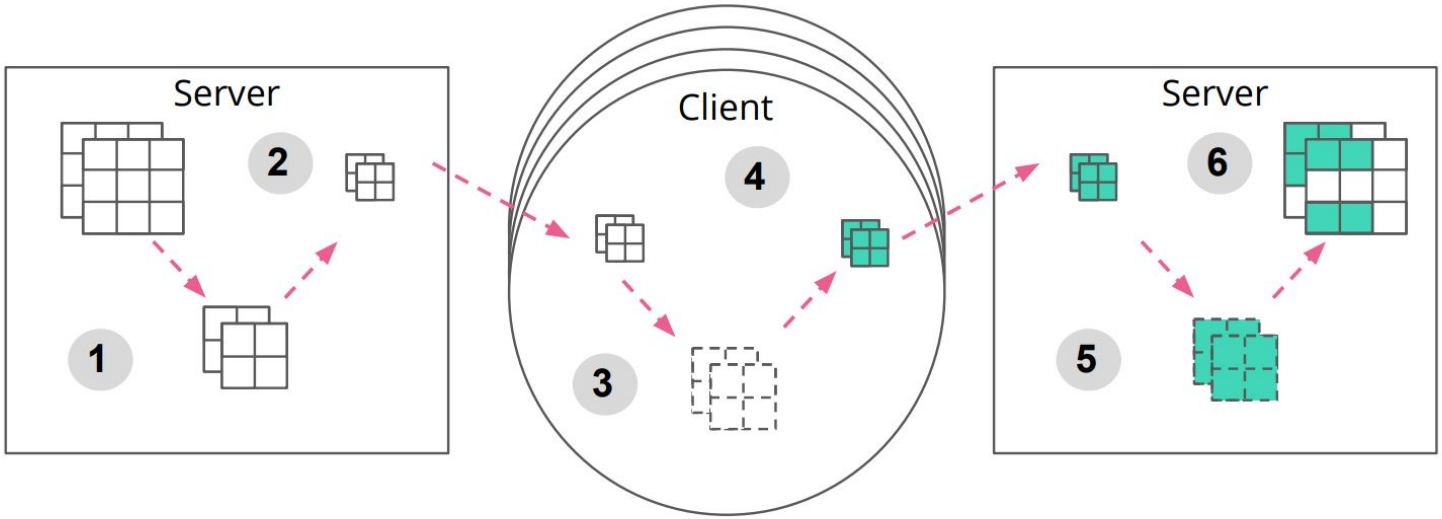


Figure 1: Combination of our proposed strategies during FL training. We reduce the size of the model to be communicated by (1) constructing a sub-model via *Federated Dropout*, and by (2) lossily compressing the resulting object. This compressed model is then sent to the client, who (3) decompresses and trains it using local data, and (4) compresses the final update. This update is sent back to the server, where it is (5) decompressed and finally, (6) aggregated into the global model.

Expanding the reach of federated learning by reducing client resource requirement

Method1: Lossy Compression

We reshape each to-be-compressed weight matrix in our model into a vector w and (1) apply a *basis transform* to it. We then (2) *subsample* and (3) *quantize* the resulting vector and finally send it through the network. Once received, we simply execute the respective inverse transformations to finally obtain a noisy version of **w**.

Expanding the reach of federated learning by reducing client resource requirement

Method2: Federated Dropout

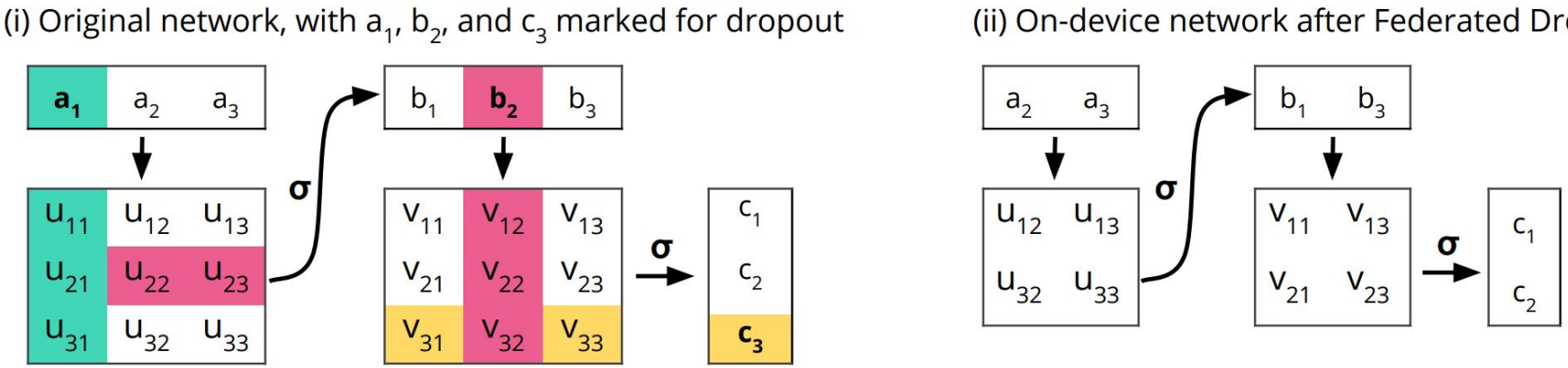


Figure 2: *Federated Dropout* applied to two fully-connected layers. Notices activation vectors $a, b = \sigma(Ua)$ and $c = \sigma(Vb)$ in (I). In this example, we randomly select exactly one activation from each layer to drop, namely a_1, b_2, and c_3, producing a sub-model with 2×2 dense matrices, as in (II).

Expanding the reach of federated learning by reducing client resource requirement

Method2: Federated Dropout

Traditional dropout: hidden units are multiplied by a random binary mask in order to drop an expected fraction of neurons during each training pass through the network. Because the mask changes in each pass, each pass is effectively computing a gradient with respect to a different sub-model. These sub-models can have different sizes (architectures) depending on how many neurons are dropped in each layer. Now, even though some units are dropped, in all implementations we are aware of, activations are still multiplied with the original weight matrices, they just have some useless rows and columns.

Expanding the reach of federated learning by reducing client resource requirement

Method2: Federated Dropout

To extend this idea to FL and realize communication and computation savings, we instead zero out a fixed number of activations at each fully-connected layer, so all possible sub-models have the same reduced architecture; see Figure 2. The server can map the necessary values into this reduced architecture, meaning only the necessary coefficients are transmitted to the client, re-packed as smaller dense matrices. The client (which may be fully unaware of the original model's architecture) trains its sub-model and sends its update, which the server then maps back to the global model2 . For convolutional layers, zeroing out activations would not realize any space savings, so we instead drop out a fixed percentage of filters.

Expanding the reach of federated learning by reducing client resource requirement

Method2: Federated Dropout

This technique brings two additional benefits beyond savings in server-to-client communication. First, the size of the client-to-server updates is also reduced. Second, the local training procedure now requires a smaller number of FLOPS per gradient evaluation, either because all matrix-multiplies are now of smaller dimensions (for fully-connected layers) or because less filters have to be applied (for convolutional ones). Thus, we reduce local computational costs.

Expanding the reach of federated learning by reducing client resource requirement

Results

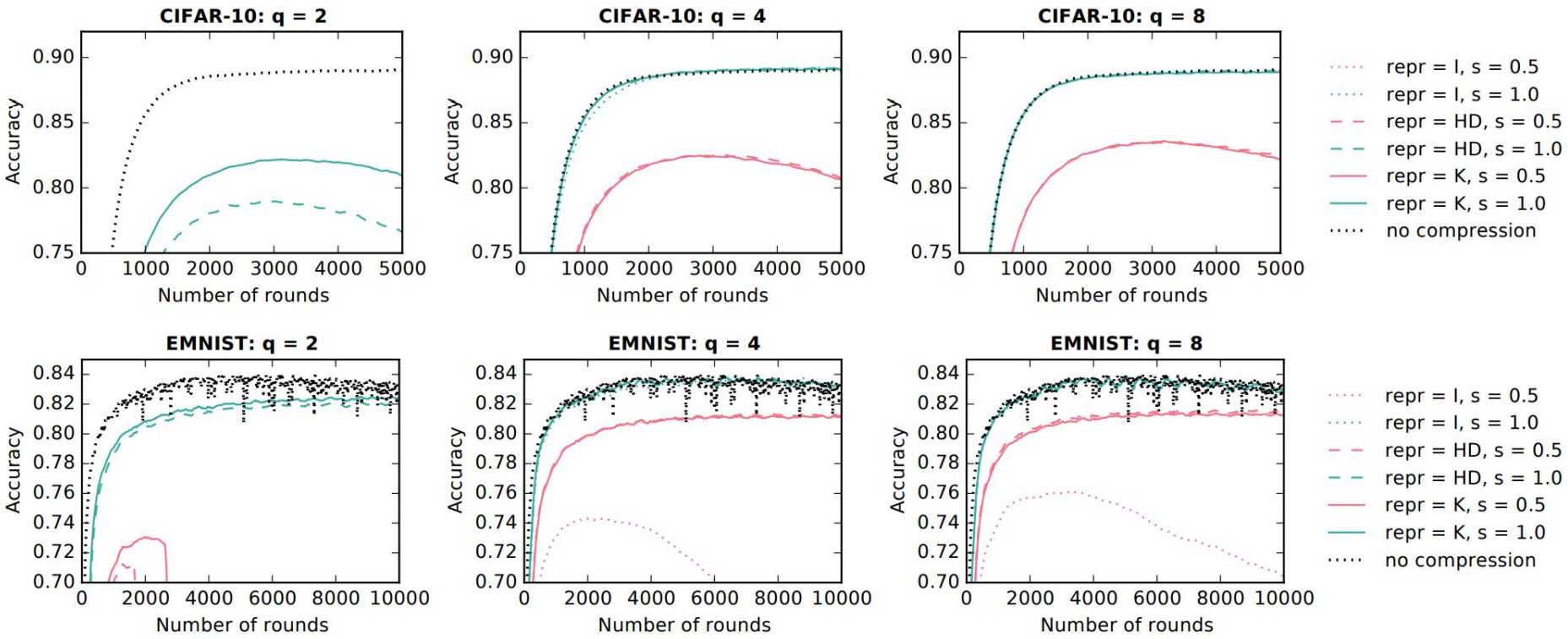


Figure 3: Effect of varying our lossy compression parameters on CIFAR-10 and EMNIST.

Expanding the reach of federated learning by reducing client resource requirement

Results

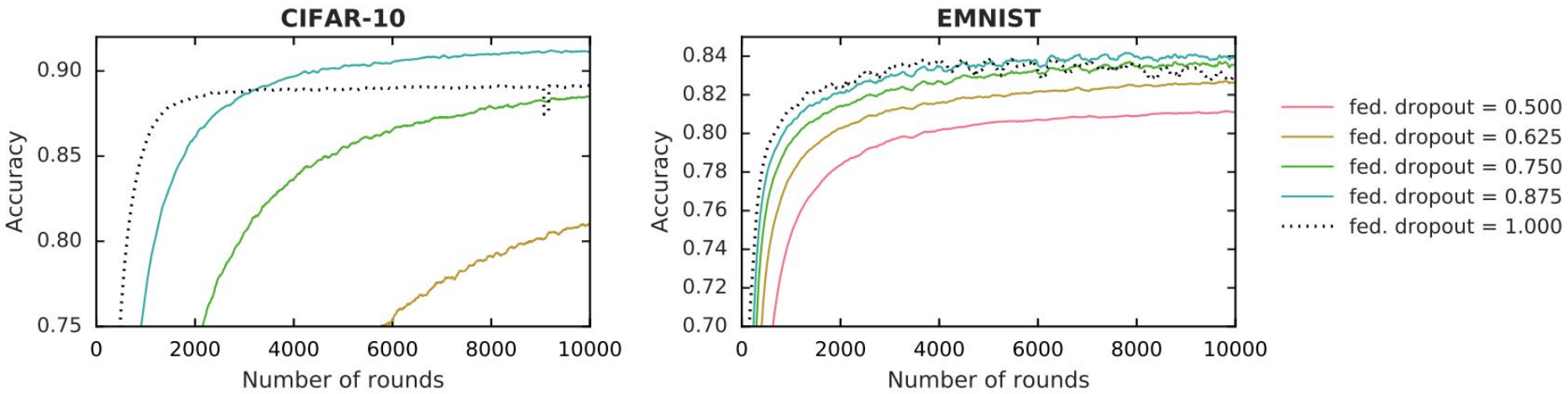


Figure 4: Results for *Federated Dropout*, varying the percentage of neurons *kept* in each layer.

Reference


Muhammad Asad, Ahmed Moustafa, and Takayuki Ito.
Fedopt: Towards communication efficiency and privacy preservation in federated learning.
Applied Sciences, 10:1–17, 04 2020.


Sebastian Caldas, Jakub Konecný, H. Brendan McMahan, and Ameet Talwalkar.
Expanding the reach of federated learning by reducing client resource requirements.
CoRR, abs/1812.07210, 2018.


Neel Guha, Ameet Talwalkar, and Virginia Smith.
One-shot federated learning.
CoRR, abs/1902.11175, 2019.