This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCOMM.2020.3003693, IEEE Communications Letters

Mix2FLD: Downlink Federated Learning After Uplink Federated Distillation With Two-Way Mixup

Seungeun Oh, †Jihong Park, Eunjeong Jeong, ‡Hyesung Kim, §Mehdi Bennis, and Seong-Lyun Kim

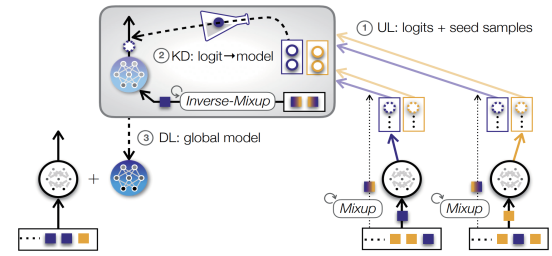
***Abstract*—This letter proposes a novel communication-efficient and privacy-preserving distributed machine learning framework, coined *Mix2FLD*. To address uplink-downlink capacity asymmetry, local model outputs are uploaded to a server in the uplink as in federated distillation (FD), whereas global model parameters are downloaded in the downlink as in federated learning (FL). This requires a model output-to-parameter conversion at the server, after collecting additional data samples from devices. To preserve privacy while not compromising accuracy, linearly mixed-up local samples are uploaded, and inversely mixed up across different devices at the server. Numerical evaluations show that Mix2FLD achieves up to 16.7% higher test accuracy while reducing convergence time by up to 18.8% under asymmetric uplink-downlink channels compared to FL.**

***Index Terms*—Distributed machine learning, on-device learning, federated learning, federated distillation, uplink-downlink asymmetry.**

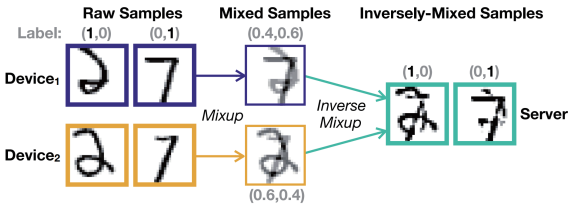
I. Introduction

User-generated local data is essential in training machine learning (ML) models for mission-critical applications, but exchanging data may violate privacy and induce huge communication overhead [1]. Federated learning (FL) is a compelling solution that collectively trains on-device ML models using their local private data [2], [3]. FL preserves data privacy, in a way that devices only upload their local model parameters to a server over wireless links, and download their average global model parameters. However, the communication efficiency of FL is problematic in deep neural network models (DNNs), since its payload sizes increase with the model sizes. The problem is aggravated in the uplink whose channel capacity is more limited by lower transmission power and bandwidth than downlink channels, i.e., uplink-downlink asymmetric channels [4]. Federated distillation (FD) resolves this problem, by exchanging model outputs [5]–[8]. Regardless of model sizes (e.g., millions of parameters in DNNs), communication payload sizes of FD are fixed as the model output dimension (e.g., 10 labels in MNIST), although FD compromises accuracy.

In order to achieve both high accuracy and communication-efficiency under uplink-downlink asymmetric channels, we propose a distributed ML framework, dubbed *Mix2FLD*. As depicted in Fig. 1, Mix2FLD is built upon two key algorithms: *federated learning after distillation (FLD)* [8] and *Mixup* data augmentation [10]. Specifically, by leveraging FLD, each device in Mix2FLD uploads its local model outputs as in FD, and downloads the model parameters as in FL, thereby coping with the uplink-downlink channel asymmetry. Between the uplink and downlink, the server runs knowledge distillation (KD) that transfers a teacher's knowledge (i.e., average outputs, see Sec II-B) to an untrained student model (i.e., a global model) [9]. This output-to-model conversion requires additional training samples collected from devices, which may violate local data privacy while incurring huge communication overhead.



(a) **Mix2FLD**: downlink federated learning (FL) & uplink federated distillation (FD) with two-way Mixup (Mix2up) seed sample collection.



(b) **Mix2up**: mixing raw samples at devices & inversely mixing them across different devices at the server (mixing ratio $\lambda = 0.4$).

Fig. 1: An illustration of (a) Mix2FLD operation and (b) Mix2up.

To preserve the data privacy with minimal communication overhead during the seed sample collection, Mix2FLD utilizes a novel *two-way Mixup algorithm (Mix2up)*, as illustrated in Fig. 1b. To hide raw samples, each device in Mix2up uploads locally superpositioned samples using Mixup [10]. Next, before running KD at the server, the uploaded mixed-up samples are superpositioned across different devices, in a way that the resulting sample labels are in the same form of the raw sample labels. This inverse-Mixup provides more realistic synthetic seed samples for KD, without restoring raw samples. Furthermore, with N_S uploaded samples, it can generate $N_I \geq N_S$ samples, enabling KD with minimal uplink cost.

Numerical simulations corroborate that Mix2FLD achieves higher test accuracy and faster training convergence than FL and FD, under both identically and independently distributed (IID) and non-IID local datasets. Furthermore, it is shown that Mix2FLD achieves higher accuracy while preserving more

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2018-0-00170, Virtual Presence in Moving Objects through 5G), Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science and ICT(NRF-2017R1A2A2A05069810), the Academy of Finland Project MISSION, SMARTER, and the 2019 EU-CHISTERA Projects LeadingEdge and CONNECT.

S. Oh, E. Jeong, and S.-L. Kim are with the School of Electrical and Electronic Engineering, Yonsei University, 120-749 Seoul, Korea (email: {seoh, ejjeong, slkim}@ramo.yonsei.ac.kr).

†J. Park is with the School of Information Technology, Deakin University, Geelong, VIC 3220, Australia (email: jihong.park@deakin.edu.au).

‡H. Kim is with Samsung Research, Samsung Electronics, Seoul, Korea (email: hye1207@gmail.com).

§M. Bennis is with the Centre for Wireless Communications, University of Oulu, 90500 Oulu, Finland (email: mehdi.bennis@oulu.fi).

TABLE I: List of Notations.

Notation	Meaning
$\mathcal{D}$	a set of devices
$\mathcal{D}^p$	a set of uploading success devices at the p-th global update
N_{ch}	# of uplink channels
N_{mod}	# of model weights
N_L	# of ground-truth labels
(N_S, N_I)	(# of mixed-up samples, # of inversely mixed-up samples)
$\mathbf{S}_d$	local dataset of the d-th device
$\mathbf{s}_d^{[i]}$	i-th sample in $\mathbf{S}_d$
• $X^{[i]}$	unlabeled sample of $\mathbf{s}_d^{[i]} = \{X^{[i]}, \mathbf{L}_n^{[i]}\}$
• $\mathbf{L}_n^{[i]}$	label vector of $\mathbf{s}_d^{[i]}$ with the n-th label as the ground-truth
$\hat{\mathbf{s}}_d^{[i,j]}$	mixed-up sample, from $\mathbf{s}_d^{[i]}$ and $\mathbf{s}_d^{[j]}$
$\tilde{\mathbf{s}}_{d,d',n}^{[i,j][i',j']}$	inversely mixed-up sample, from $\hat{\mathbf{s}}_d^{[i,j]}$ and $\hat{\mathbf{s}}_{d'}^{[i',j']}$
$\mathbf{G}_{\mathsf{mod}}^p$	global weight vector at the p-th global update
• $\mathbf{w}_d^{(k)}$	local weight vector at the k-th iteration
$\mathbf{G}_{\mathsf{out},n}^p$	global output vector with the n-th ground-truth label
• $\mathbf{F}_{d,n}^{[i_k]}$	local output vector (i.e., softmax logits) of the sample i_k
• $\bar{\mathbf{F}}_{d,n}^p$	local average output vector with the n-th ground-truth label
K	# of local iterations per global update
K_s	# of output-to-parameter converting iterations per global update
T^y	# of time slots for uploading or downloading B^y bits

data privacy, compared to FLD using only Mixup (MixFLD), highlighting the importance of Mix2up.

II. System Model

This section describes our baseline distributed ML architecture and operations, followed by communication channel models. The network under study comprises a set $\mathcal{D}$ of devices connected to a single server through wireless links. Following a data-parallel distributed ML architecture [1], every device owns its local private dataset and an on-device ML model having N_{mod} weights. The d-th device has a local dataset $\mathbf{S}_d$ of samples, in which the i-th sample $\mathbf{s}_d^{[i]} = \{X^{[i]}, \mathbf{L}_n^{[i]}\}$ comprises a pair of an unlabeled sample $X^{[i]}$ and its label vector $\mathbf{L}_n^{[i]} = \{\ell_1^{[i]}, \ell_2^{[i]}, \cdots, \ell_{N_L}^{[i]}\}$. The label vector's element $\ell_n^{[i]}$ equals 1 if the n-th label is the ground-truth, and 0 otherwise.

With $\mathbf{S}_d$, each device collaboratively trains its model so as to predict the labels of unlabeled data samples in a multi-class classification task. These distributed ML operations are divided into local updates at devices and global updates at the server, as detailed next under FL and FD.

A. FL

For each global update, every device updates its local model weights by running K iterations of the stochastic gradient descent algorithm (SGD). At the k-th local iteration, the d-th device randomly selects the i_k-th sample, and updates its *local weight vector* $\mathbf{w}_d^{(k)}$ with a constant learning rate η as:

$$\mathbf{w}_d^{(k+1)} = \mathbf{w}_d^{(k)} - \eta \nabla \phi(\mathbf{F}_{d,n}^{[i_k]}, \mathbf{L}_n^{[i_k]} | \mathbf{w}_d^{(k)}), \tag{1}$$

by calculating the gradient of a cross-entropy loss function $\phi(\mathbf{F}_{d,n}^{[i_k]}, \mathbf{L}_n^{[i_k]} | \mathbf{w}_d^{(k)}) = -\sum_{m=1}^{N_L} \ell_m^{[i_k]} \log F_m^{[i_k]}$. The term $\mathbf{F}_{d,n}^{[i_k]}$ is the *local output vector* $\mathbf{F}_{d,n}^{[i_k]} = \{F_1^{[i_k]}, F_2^{[i_k]}, \cdots, F_{N_L}^{[i_k]}\}$ implying the prediction distribution over N_L labels when the n-th label is the ground-truth. The elements of $\mathbf{F}_{d,n}^{[i_k]}$ are softmax normalized logits at the model's last layer, satisfying $\sum_{m=1}^{N_L} F_m^{[i_k]} = 1$ with $F_m^{[i_k]} \in [0,1]$ $\forall m$.

After K local iterations, following [2], the d-th device in FL uploads its *latest weight vector* $\mathbf{w}_d^{(K)}$ to the server over a wireless link. A set $\mathcal{D}^p$ of devices can successfully upload the weight vectors at the p-th global update, depending on the channel conditions that will be elaborated in Sec. II-C. By taking a weighted average proportional to the number of samples $|\mathcal{D}^p|$ devices have, the server produces the *global weight vector* $\mathbf{G}_{\mathsf{mod}}^p = \sum_{d\in\mathcal{D}^p} |\mathbf{S}_d| \mathbf{w}_d^{(K)} / \sum_{d\in\mathcal{D}^p} |\mathbf{S}_d|$ that is downloaded by each device. Finally, the d-th device replaces $\mathbf{w}_d^{(K)}$ with $\mathbf{G}_{\mathsf{mod}}^p$, and continues its local updates in (1) until the $(p+1)$-th global update. These operations are iterated until $|\mathbf{G}_{\mathsf{mod}}^p - \mathbf{G}_{\mathsf{mod}}^{p-1}| / |\mathbf{G}_{\mathsf{mod}}^{p-1}| < \varepsilon$ is satisfied, for a constant $\varepsilon > 0$.

B. FD

At the p-th global update, following [5], the d-th device in FD uploads N_L *local average output vectors*, produced by averaging the local output vectors $\{\mathbf{F}_{d,n}^{[i_k]}\}$ during K local SGD iterations, separately for each ground-truth label. For the n-th ground-truth label (i.e., $\ell_n^{[i_k]} = 1$), the local average output vector $\bar{\mathbf{F}}_{d,n}^p$ is given as:

$$\bar{\mathbf{F}}_{d,n}^p = \sum_{k=1}^{K} \mathbb{1}\left(\ell_n^{[i_k]} = 1\right) \mathbf{F}_d^{[i_k]} / \sum_{k=1}^{K} \mathbb{1}\left(\ell_n^{[i_k]} = 1\right), \tag{2}$$

where $\mathbb{1}(A)$ becomes 1 if A is true, and 0 otherwise. By averaging $\{\bar{\mathbf{F}}_{d,n}^p\}$ across $|\mathcal{D}^p|$ devices, the server generates the *global average output vector* $\mathbf{G}_{\mathsf{out},n}^p = \sum_{d\in\mathcal{D}^p} \bar{\mathbf{F}}_{d,n}^p / |\mathcal{D}^p| = \{G_1, G_2, \cdots, G_{N_L}\}$ that is downloaded by each device.

Next, until the $(p+1)$-th global update, the d-th device updates its local weight vector $\mathbf{w}_d^{(k)}$ using SGD with KD as:

$$\mathbf{w}_d^{(k+1)} = \mathbf{w}_d^{(k)} - \eta \nabla \Big(\phi(\mathbf{F}_{d,n}^{[i_k]}, \mathbf{L}_n^{[i_k]} | \mathbf{w}_d^{(k)}) + \beta \psi(\mathbf{F}_{d,n}^{[i_k]}, \mathbf{G}_{\mathsf{out},n}^p) \Big), \tag{3}$$

with a constant $\beta > 0$. In contrast to (1), this includes a *distillation regularizer* $\psi(\mathbf{F}_{d,n}^{[i_k]}, \mathbf{G}_{\mathsf{out},n}^p) = \sum_{m=1}^{N_L} G_m \log F_m^{[i_k]}$ that measures the gap between $\mathbf{F}_{d,n}^{[i_k]}$ and $\mathbf{G}_{\mathsf{out},n}^p$ using cross-entropy. If this knowledge gap is negligible, the device's weight is updated based on its own prediction, and otherwise perturbed proportionally to the gap. These operations continue until $|\mathbf{G}_{\mathsf{out},n}^p - \mathbf{G}_{\mathsf{out},n}^{p-1}| / |\mathbf{G}_{\mathsf{out},n}^{p-1}| < \varepsilon$ is satisfied for all n.

C. Wireless Channel Model

At each global update, we consider uplink unicast and downlink multicast transmissions. In the uplink, the server allocates equal bandwidth $W^{\mathrm{up}} = W N_{ch} / |\mathcal{D}|$ to each device for frequency division multiple access (FDMA), whereas in the downlink it utilizes the entire bandwidth $W^{\mathrm{dn}} = W$. Let the superscript $y = \{\mathrm{up}, \mathrm{dn}\}$ identify uplink and downlink. With the transmission power P^y and the distance r_d from the d-th device to the server, the received signal-to-noise ratio (SNR) in either uplink or downlink at the t-th time slot is $\mathsf{SNR}_{d,t}^y = P^y h_{d,t} {r_d}^{-\alpha} / (W^y N_0)$, where N_0 is the noise power spectral density, and $\alpha \geq 2$ denotes the path loss exponent. Following Rayleigh block fading channels, the term $h_{d,t}$ is an exponential random variable with unitary mean, independent and identically distributed (IID) across different devices and time slots.

For a target SNR $\theta^y > 0$, each received signal is successfully decoded if $\mathsf{SNR}_{d,t}^y \geq \theta^y$. During T time slots, the received B_{RX}^y bits is thereby given as:

$$B_{\mathrm{RX}}^y(T) = \tau \sum_{t=1}^{T} \mathbb{1}(\mathsf{SNR}_{d,t}^y \geq \theta^y) W^y \log_2(1+\theta^y), \tag{4}$$

where τ is the channel coherence time identically set as the unit time slot. The latency T^y slots (or τT^y seconds) for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCOMM.2020.3003693, IEEE Communications Letters

Algorithm 1 FLD with Mix2up (**Mix2FLD**)

Require: $\{\mathbf{S}_d\}$ with $d \in \mathcal{D}$, $\lambda \in (0,1)$
while $|\mathbf{G}^p_{\text{out},n} - \mathbf{G}^{p-1}_{\text{out},n}|/|\mathbf{G}^{p-1}_{\text{out},n}| \geq \varepsilon$ **do**
Device $d \in \mathcal{D}$: ▷ *Output upload*
  if $p=1$ **generates** $\{\hat{\mathbf{s}}_d^{[i,j]}\}$ via (6) **end if** ▷ *Mixup*
  updates $\mathbf{w}_d^{(k)}$ in (1) and $\bar{\mathbf{F}}^p_{d,n}$ in (2) for K iterations
  unicasts $\{\bar{\mathbf{F}}^p_{d,n}\}$ (with $\{\hat{\mathbf{s}}_d^{[i,j]}\}$ **if** $p=1$) to the server
Server: ▷ *Output-to-model conversion*
  if $p=1$ **generates** $\{\tilde{\mathbf{s}}^{[i,j][i',j']}_{d,d',n}\}$ via (7) **end if** ▷ *Inverse-Mixup*
  computes $\mathbf{G}^p_{\text{out},n}$
  updates $\mathbf{w}_s^{(k)}$ via (5) for K_s iterations
  broadcasts $\mathbf{G}^p_{\text{mod}} = \mathbf{w}_s^{(K_s)}$ to all devices
$p \leftarrow p+1$
Device $d \in \mathcal{D}$ **substitutes** $\mathbf{w}_d^{(0)}$ with $\mathbf{G}^p_{\text{mod}}$ ▷ *Model download*
end while

uploading or downloading B^y bits is the minimum T that satisfies $B^y_{\text{RX}}(T) \geq B^y$. In order to avoid unbounded latency, the server allocates up to $T_{\max}$ time slots equally to the uplink and downlink. A latency outage occurs when $T^y > T_{\max}$, incurring a straggling device.

In FL, the payload is the model weights, resulting in $B^{\text{up}} = B^{\text{dn}} = b_{\text{mod}} N_{\text{mod}}$ bits, where b_{mod} is each weight size determined by its arithmetic precision. In FD, N_L output vectors are exchanged each of which has N_L elements, leading to $B^{\text{up}} = B^{\text{dn}} = b_{\text{out}} {N_L}^2$ bits, where b_{out} denotes each output size.

III. Mix2FLD: Federated Learning After Distillation With Two-Way Mixup

In this section, we propose the idea of FLD and its two implementations, MixFLD and Mix2FLD. Leveraging the Mixup algorithm [10], MixFLD enables FLD while preserving data privacy during its seed sample collection. Mix2FLD integrates our novel inverse-Mixup algorithm into MixFLD, further improving accuracy.

A. FLD

FLD aims to address asymmetric uplink-downlink channel capacity. As shown in Fig. 1a, at the p-th global update, the d-th device uploads N_L local average output vectors $\{\bar{\mathbf{F}}^p_{d,n}\}$, thereby constructing the global average output vector $\mathbf{G}^p_{\text{out},n}$ at the server, as in FD. Then, the device downloads the global weight vector $\mathbf{G}^p_{\text{mod}}$ as in FL. The problem is that the server in FLD lacks $\mathbf{G}^p_{\text{mod}}$, calling for converting $\mathbf{G}^p_{\text{out},n}$ into $\mathbf{G}^p_{\text{mod}}$.

Output-to-Model Conversion. The key idea is to transfer the knowledge in $\mathbf{G}^p_{\text{out},n}$ to a global model having the weight vector $\mathbf{G}^p_{\text{mod}}$. To enable this, at the beginning (i.e., $p=1$), each device uploads N_s seed samples randomly selected from its local dataset. By feeding the collected $|\mathcal{D}| \cdot N_s$ seed samples, as done in (3), the server runs K_s iterations of SGD with KD, thereby updating the global model's weight vector $\mathbf{w}_s^{(k)}$ as:

$$\mathbf{w}_s^{(k+1)} = \mathbf{w}_s^{(k)} - \eta \nabla \Big(\phi(\mathbf{F}^{[i_k]}_{s,n}, \mathbf{L}^{[i_k]}_n | \mathbf{w}_s^{(k)}) + \beta \psi(\mathbf{F}^{[i_k]}_{s,n}, \mathbf{G}^p_{\text{out},n}) \Big), \quad (5)$$

where $\mathbf{F}^{[i_k]}_{s,n}$ is the global model's output vector if the n-th label is the ground-truth. Finally, the server yields $\mathbf{G}^p_{\text{mod}} = \mathbf{w}_s^{(K_s)}$ that is downloaded by every device. The remaining operations follow the same procedure of FL. In FLD, $B^{\text{up}} = b_{\text{out}} {N_L}^2 + \mathbb{1}\{p=1\} b_s N_s$ bits, and $B^{\text{dn}} = b_{\text{mod}} N_{\text{mod}}$ bits, where b_s is the size of each sample.

B. MixFLD: FLD + Mixup

The aforementioned FLD operations include seed sample collection that may violate local data privacy. To mitigate this problem, MixFLD applies the Mixup to the sample collection procedure of FLD, as follows.

Mixup Before Collection. Before uploading the seed samples, the d-th device randomly selects two different raw samples $\mathbf{s}_d^{[i]}$ and $\mathbf{s}_d^{[j]}$ having different labels, i.e., $\mathbf{L}_n^{[i]} \neq \mathbf{L}_m^{[j]}$ with $m \neq n$ and $i \neq j$. With a mixing ratio $\lambda \in (0, 0.5)$ given identically for all devices, the device linearly combines these two samples (see Fig. 1b), thereby generating a mixed-up sample $\hat{\mathbf{s}}_d^{[i,j]}$ as:

$$\hat{\mathbf{s}}_d^{[i,j]} = \lambda \mathbf{s}_d^{[i]} + (1-\lambda) \mathbf{s}_d^{[j]}. \quad (6)$$

In this way, the device uploads N_s mixed-up samples to the server, and the rest of procedures follow FLD.

C. Proposed. Mix2FLD: MixFLD + Inverse-Mixup

It is observed that MixFLD significantly distorts the seed samples, achieving lower accuracy than FD, in our numerical evaluations in Sec. IV. To ensure not only data privacy but also high accuracy, we propose Mix2FLD that integrates our novel inverse-Mixup algorithm into MixFLD.

For the sake of explanation, we hereafter focus on a two-device setting, where devices d and d' independently mix up the following two raw samples having symmetric labels.

- Device d: $\mathbf{s}_d^{[i]}$ with $\mathbf{L}_1^{[i]} = \{1,0\}$ and $\mathbf{s}_d^{[j]}$ with $\mathbf{L}_2^{[j]} = \{0,1\}$
- Device d': $\mathbf{s}_{d'}^{[i']}$ with $\mathbf{L}_2^{[i']} = \{0,1\}$ and $\mathbf{s}_{d'}^{[j']}$ with $\mathbf{L}_1^{[j']} = \{1,0\}$

According to (6), the mixed-up samples $\hat{\mathbf{s}}_d^{[i,j]}$ and $\hat{\mathbf{s}}_{d'}^{[i',j']}$ have the *soft labels* $\{\lambda, 1-\lambda\}$ and $\{1-\lambda, \lambda\}$, respectively, in contrast to the *hard labels* $\{1,0\}$ and $\{0,1\}$ of raw samples.

Inverse-Mixup After Collection. Before training the global model using (5), the sever in Mix2FLD converts the soft labels back into hard labels. To this end, we propose *inverse-Mixup* that linearly combines N mixed-up samples such that the resulting sample has a hard label. For the case of $N=2$, as shown in Fig. 1b, with the above-mentioned symmetric setting, the server combines $\hat{\mathbf{s}}_d^{[i,j]}$ and $\hat{\mathbf{s}}_{d'}^{[i',j']}$, such that the resulting $\tilde{\mathbf{s}}^{[i,j][i',j']}_{d,d',n}$ has the n-th converted hard label as the ground-truth. This is described as:

$$\tilde{\mathbf{s}}^{[i,j][i',j']}_{d,d',n} = \hat{\lambda} \hat{\mathbf{s}}_d^{[i,j]} + (1-\hat{\lambda}) \hat{\mathbf{s}}_{d'}^{[i',j']}. \quad (7)$$

The inverse mixing ratio $\hat{\lambda}$ for $N \geq 2$ is chosen in the following way.

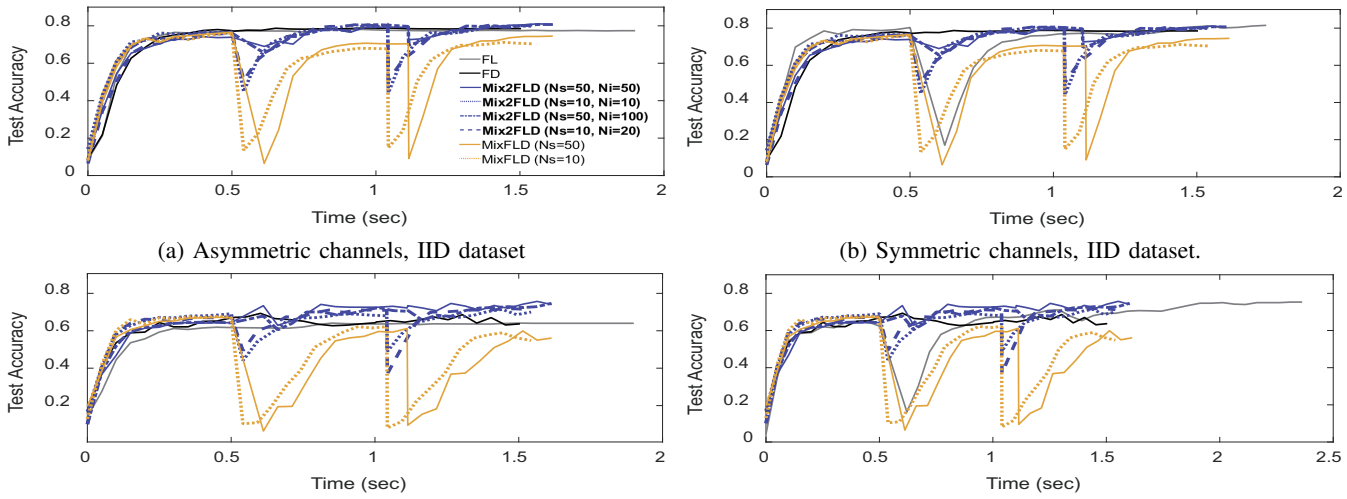
Proposition 1. *When N raw samples are combined with the mixing ratios $(\lambda_1, \lambda_2, \ldots, \lambda_N)$, the inverse mixing ratios $(\hat{\lambda}_{1,n}, \hat{\lambda}_{2,n}, \ldots, \hat{\lambda}_{N,n})$ that make an inversely-mixup sample has the n-th label as the ground-truth are given by solving the following equation.*

$$\begin{bmatrix} \hat{\lambda}_{1,1} & \hat{\lambda}_{1,2} & \ldots & \hat{\lambda}_{1,N} \\ \hat{\lambda}_{2,1} & \hat{\lambda}_{2,2} & \ldots & \hat{\lambda}_{2,N} \\ \vdots & \vdots & \vdots & \vdots \\ \hat{\lambda}_{N,1} & \hat{\lambda}_{N,2} & \ldots & \hat{\lambda}_{N,N} \end{bmatrix} = \begin{bmatrix} \lambda_1 & \lambda_2 & \ldots & \lambda_N \\ \lambda_2 & \lambda_3 & \ldots & \lambda_1 \\ \vdots & \vdots & \vdots & \vdots \\ \lambda_N & \lambda_1 & \ldots & \lambda_{N-1} \end{bmatrix}^{-1}, \quad (8)$$

where $\sum_{d=1}^N \lambda_d = 1$.

Proof: First, consider $N=2$. Suppose the target hard label is $\{1,0\}$, i.e., $n=1$. Applying $\{1,0\}$ to the LHS of (7) and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCOMM.2020.3003693, IEEE Communications Letters



(a) Asymmetric channels, IID dataset

(b) Symmetric channels, IID dataset.

(c) Asymmetric channels, Non-IID dataset.

(d) Symmetric channels, Non-IID dataset.

Fig. 2: Learning curve of a randomly selected device in Mix2FLD, compared to FL, FD, and MixFLD, under asymmetric ($P^{\text{up}} = 23$ dBm, $P^{\text{dn}} = 40$ dBm) and symmetric ($P^{\text{up}} = P^{\text{dn}} = 40$ dBm) channels, when $\lambda = 0.1$ with IID and non-IID datasets.

$\{\lambda, 1-\lambda\}$ and $\{1-\lambda, \lambda\}$ of $\hat{\mathbf{s}}_d^{[i,j]}$ and $\hat{\mathbf{s}}_{d'}^{[i',j']}$ to the RHS of (7) yields two equations.

$$1 = \hat{\lambda}\lambda + (1-\hat{\lambda})(1-\lambda) \tag{9}$$

$$0 = \hat{\lambda}(1-\lambda) + (1-\hat{\lambda})\lambda \tag{10}$$

Solving these equations yields the desired $\hat{\lambda}$. By induction, this can be generalized to $N \geq 2$, completing the proof. ■

Hereafter, for the sake of convenience, we fix N to 2. By alternating $\hat{\lambda}$ with $n = 1$ and 2, inversely mixing up two mixed-up samples $\hat{\mathbf{s}}_d^{[i,j]}$ and $\hat{\mathbf{s}}_{d'}^{[i',j']}$ yields two inversely mixed-up samples $\tilde{\mathbf{s}}_{d,d',1}^{[i,j][i',j']}$ and $\tilde{\mathbf{s}}_{d,d',2}^{[i,j][i',j']}$. The server generates N_I inversely mixed-up samples by pairing two samples with symmetric labels among N_S mixed-up samples. By nature, inverse-Mixup is a data augmentation scheme, so N_I can be larger than N_S. Finding the optimal N_I that achieves the highest accuracy with minimal memory usage could be an interesting topic for future work.

Note that none of the raw samples are identical to inversely mixed-up samples. To ensure this, inverse-Mixup is applied only for the seed samples uploaded from different devices, thereby preserving data privacy. The overall operation of Mix2FLD is summarized in Algorithm 1.

IV. Numerical Evaluation and Discussions

In this section, we numerically evaluate the performance of Mix2FLD compared to FL, FD, and MixFLD, in terms of the test accuracy and convergence time of a randomly selected reference device, under different data distributions (IID and non-IID) and uploaded/generated seed sample configurations ($(N_S, N_I) \in \{(10,10),(10,20),(50,50),(50,100)\}$). The convergence time includes communication delays $\tau(T^{\text{up}} + T^{\text{dn}})$ seconds during the uplink and downlink (see Sec. II-C), as well as the computing delays of devices and the server, which are measured using tic-toc elapsed time.

Every device has a 3-layer convolutional neural network model (2 convolutional layers, 1 fully-connected layer) having $N_{\text{mod}} = 12{,}544$. The server's global model follows the same architecture. The model weight and output parameter sizes are given identically as $b_{\text{mod}} = b_{\text{out}} = 32$ bits.

Each device owns its local MNIST dataset with $N_L = 10$ and $|\mathbf{S}_d| = 500$. For the IID case, every label has the same number of samples. For the non-IID case, randomly selected two labels have 2 samples respectively, while each of the other labels has 62 samples. Each sample size is given as $b_s = 6{,}272$ bits (8 bits $\times$ (28×28) pixels).

Other simulation parameters are given as: $|\mathcal{D}| = 10$, $K = 6{,}400$ iterations, $K_{\text{KD}} = 3{,}200$ iterations, $\eta = 0.01$, $\varepsilon = 0.05$, β=0.01, $N_{ch} = 2$, $W = 10$ MHz, $P^{\text{up}} = 23$ dBm, $P^{\text{dn}} = 40$ dBm, $r_d = 1$ km, $\alpha = 4$, $N_0 = -174$ dBm/Hz, $\theta^{\text{up}} = \theta^{\text{dn}} = 3$, $\tau = 1$ ms, and $T_{\max} = 100$ ms.

Impact of Channel Conditions. Fig. 2 shows that Mix2FLD achieves the highest accuracy with moderate convergence under asymmetric channel conditions. Compared to FL uploading model weights, Mix2FLD's model output uploading reduces the uplink payload size by up to 42.4x. Under asymmetric channels with the limited uplink capacity (Figs. 2a and c), this enables more frequent and successful uploading, thereby achieving up to 16.7% higher accuracy and 1.2x faster convergence. Compared to FD, Mix2FLD leverages the high downlink capacity for downloading the global model weights, which often provides higher accuracy than downloading model outputs as reported in [5]. In addition, the global information of Mix2FLD is constructed by collecting seed samples and reflecting the global data distribution, rather than by simply averaging local outputs as used in FD. Thereby, Mix2FLD achieves up to 17.3% higher accuracy while taking only 2.5% more convergence time than FD. Under symmetric channels, FL achieves the highest accuracy. Nevertheless, Mix2FLD still converges 1.9x faster than FL, thanks to its smaller uplink payload sizes and more frequent updates.

Fluctuation of Test Accuracy. FL, MixFLD, and Mix2FLD have instantaneous accuracy drop in global update. After downloading the global information, a noise reflecting global data distributions is inserted into local models, leading to a drastic decrease in test accuracy. This accuracy drop is gradually recovered during local updates, and finally higher accuracy can be achieved than before the noise insertion. In

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCOMM.2020.3003693, IEEE Communications Letters

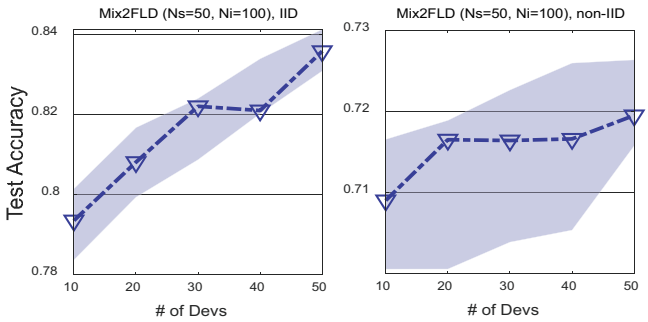


Fig. 3: Test accuracy distribution of Mix2FLD w.r.t the number of devices, under symmetric channels with IID and non-IID datasets.

FD, a noise is inserted for every training sample, and partially reflected as an additional loss function, resulting in smaller accuracy drops.

Impact of the Number of Devices. Fig. 3 shows the scalability of Mix2FLD, under both IID and non-IID data distributions. When the number of devices is increased from 10 to 50, the average of test accuracy increases by 5.7% and variance decreases by 50% with IID dataset. In the non-IID dataset, the test accuracy gain is smaller than that of the IID dataset, while having the same tendency.

Impact of Mix2up. Fig. 2c and Fig. 2d corroborate that Mix2FLD is particularly effective in coping with non-IID data. In our non-IID datasets, samples are unevenly distributed, and locally trained models become more biased, degrading accuracy compared to IID datasets in Fig. 2a and Fig. 2c. This accuracy loss can partly be restored by additional global training (i.e., output-to-model conversion) that reflects the entire dataset distribution using few seed samples. While preserving data privacy, MixFLD attempts to realize this idea. However, as observed in Fig. 2d, MixFLD fails to achieve high accuracy as its mixed-up samples inject too much noise into the global training process. Mix2FLD resolves this problem by utilizing inversely mixed up samples, reducing unnecessary noise. Thanks to its incorporating the data distribution, even under symmetric channels (Fig. 2d), Mix2FLD achieves the accuracy as high as FL.

Latency, Privacy, and Accuracy Tradeoffs. For all cases in Fig. 2, in Mix2FLD and MixFLD, reducing the seed sample amount ($N_s = 10$) provides faster convergence time in return for compromising accuracy, leading to a *latency-accuracy* tradeoff. Furthermore, in Fig. 2, even if N_S is the same, when N_I is large, the accuracy increases up to 1.7%. Such data augmentation effect of inverse-Mixup enables Mix2FLD to effectively increase accuracy without additional latency. Next, we validate the data privacy guarantees of Mixup and Mix2up. This is evaluated using *sample privacy*, given as the minimum similarity between a mixed-up sample and its raw sample: $\log(\min\{||\hat{\mathbf{s}}_d^{[i,j]} - \mathbf{s}_d^{[i]}||, ||\hat{\mathbf{s}}_d^{[i,j]} - \mathbf{s}_d^{[j]}||\})$ according to [11], [12]. Table II shows that Mixup ($\lambda > 0$) with a single device preserves more sample privacy than the case without Mixup ($\lambda = 0$). Table III illustrates that Mix2up with two devices preserves higher sample privacy than Mixup thanks to its additional (inversely) mixing up the seed samples across devices. It also shows that each inversely mixed-up sample does not resemble its raw sample but an arbitrary sample having the same ground-truth label. Both Tables II and III show that the mixing ratio λ closer to 0.5 (i.e., equally mixing up two samples) ensures higher sample privacy, which may require compromising more accuracy. Investigating the *privacy-accuracy* tradeoff is deferred to future work.

TABLE II: Sample privacy, *Mixup* (N_s=100).

Dataset	**Sample Privacy** Under Mixing Ratio λ					
	λ = 0.001	0.1	0.2	0.3	0.4	0.499
MNIST	2.163	4.465	5.158	5.564	5.852	**6.055**
FMNIST	1.825	4.127	4.821	5.226	5.514	**5.717**
CIFAR-10	2.582	4.884	5.577	5.983	6.270	**6.473**
CIFAR-100	2.442	4.744	5.438	5.843	6.131	**6.334**

TABLE III: Sample privacy, *Mix2up* (N_s=100).

Dataset	**Sample Privacy** Under Mixing Ratio λ					
	λ = 0.001	0.1	0.2	0.3	0.4	0.499
MNIST	2.557	4.639	5.469	6.140	7.007	**9.366**
FMNIST	2.196	4.568	5.410	6.143	6.925	**9.273**
CIFAR-10	2.824	5.228	6.076	6.766	7.662	**10.143**
CIFAR-100	2.737	5.151	6.050	6.782	7.652	**10.104**

V. Concluding Remarks

In this letter, we proposed Mix2FLD that copes with asymmetric uplink-downlink channel capacities, while preserving data privacy. Numerical evaluations corroborated its effectiveness in terms of accuracy and convergence time, under supervised learning in the MNIST classification task. Applying Mix2up to other distributed learning scenarios could be an interesting topic for future research. Also, extending this idea to distributed reinforcement learning by leveraging the proxy experience memory method as in [6] as well as the convergence analysis of Mix2FLD is left to future work.

References

[1] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless Network Intelligence at the Edge," *Proc. IEEE,* vol. 107, no. 11, pp. 2204-2239, Nov. 2019.

[2] P. Kairouz, et al., "Advances and Open Problems in Federated Learning," [Online]. ArXiv preprint: http://arxiv.org/abs/1912.04977, Dec. 2019.

[3] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated Machine Learning: Concept and Applications," *ACM Trans. Intell. Syst. Technol.,* vol. 10, no.2, pp. 1-19, Jan. 2019.

[4] J. Park, S.-L. Kim, and J. Zander, "Tractable Resource Management with Uplink Decoupled Millimeter-Wave Overlay in Ultra-Dense Cellular Networks," *IEEE Trans. Wireless Commun.,* vol. 15, no.6, pp. 4362–4379, Jun. 2016.

[5] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-Efficient On-Device Machine Learning: Federated Distillation and Augmentation under Non-IID Private Data," [Online]. ArXiv preprint: http://arxiv.org/abs/1811.11479, Nov. 2019.

[6] H. Cha, J. Park, H. Kim, S.-L. Kim, and M. Bennis, "Federated Reinforcement Distillation with Proxy Experience Memory," [Online]. ArXiv preprint: http://arxiv.org/abs/1907.06536, Jul. 2019.

[7] J. H. Ahn, O. Simeone, and J. Kang, "Wireless Federated Distillation for Distributed Edge Learning with Heterogeneous Data," in *Proc. IEEE Int. Symp. Pers., Indoor and Mobile Radio Commun.,* Sep. 2019.

[8] J. Park, et al., "Distilling On-Device Intelligence at the Network Edge," [Online]. Arxiv preprint: http://arxiv.org/abs/1908.05895, Aug. 2019.

[9] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," [Online]. Arxiv preprint: http://arxiv.org/abs/1503.02531, Mar. 2015.

[10] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond Empirical Risk Minimization," [Online]. Arxiv preprint: http://arxiv.org/abs/1710.09412, Oct. 2017.

[11] P. Mair, "Multidimensional Scaling," *Modern Psychometrics with R,* pp. 257–287, Springer, Sep. 2018.

[12] E. Jeong, S. Oh, J. Park, H. Kim, M. Bennis, and S.-L. Kim, "Multi-hop Federated Private Data Augmentation with Sample Compression," [Online]. Arxiv preprint: http://arxiv.org/abs/1907.06426, Jul. 2019.