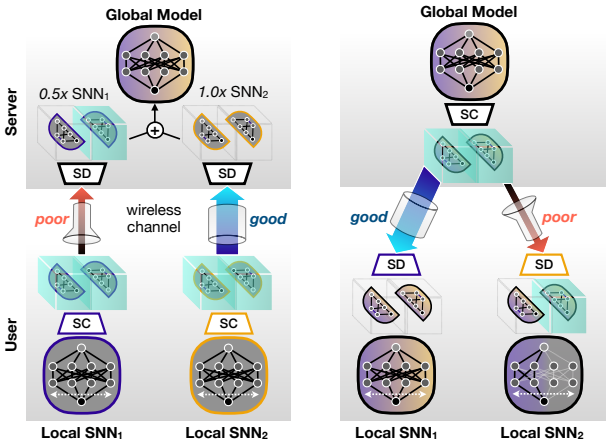
Communication and Energy Efficient Slimmable Federated Learning via Superposition Coding and Successive Decoding

Hankyul Baek [1] **Won Joon Yun** [1] **Soyi Jung** [1] **Jihong Park** [2] **Mingyue Ji** [3] **Joongheon Kim** [1] **Mehdi Bennis** [4]

Abstract

Mobile devices are indispensable sources of big data. Federated learning (FL) has a great potential in exploiting these private data by exchanging locally trained models instead of their raw data. However, mobile devices are often energy limited and wirelessly connected, and FL cannot cope flexibly with their heterogeneous and time-varying energy capacity and communication throughput, limiting the adoption. Motivated by these issues, we propose a novel energy and communication efficient FL framework, coined *SlimFL*. To resolve the heterogeneous energy capacity problem, each device in SlimFL runs a width-adjustable *slimmable neural network (SNN)*. To address the heterogeneous communication throughput problem, each full-width (1.0x) SNN model and its half-width (0.5x) model are *superposition-coded* before transmission, and *successively decoded* after reception as the 0.5x or 1.0x model depending on the channel quality. Simulation results show that SlimFL can simultaneously train both 0.5x and 1.0x models with reasonable accuracy and convergence speed, compared to its vanilla FL counterpart separately training the two models using 2x more communication resources. Surprisingly, SlimFL achieves even higher accuracy with lower energy footprints than vanilla FL for poor channels and non-IID data distributions, under which vanilla FL converges slowly.



(a) Uplink. (b) Downlink.

Figure 1. Slimmable federated learning (SlimFL) in the (a) uplink and (b) downlink, with the superposition coding (SC) and successive decoding (SD) of the slimmable neural networks (SNNs).

1. Introduction

Federated learning (FL) is a promising solution to enable high-quality on-device learning at mobile devices such as phones (Hard et al., 2018), cars (Samarakoon et al., 2018), and drones (Shiri et al., 2020). Each of these devices has only a limited amount of local data, and FL can overcome the lack of local model training samples by exchanging and aggregating the local models of different devices. To reach its full potential, it is essential to scale up the range of federating devices that are often wirelessly connected while having heterogeneous levels of available energy (Kairouz et al., 2019; Park et al., 2021). This mandates addressing the following interrelated energy and wireless communication problems.

Heterogeneous Energy Capacity Problem. Different devices have heterogeneous levels of available energy. Low-energy devices are likely to run small models, whereas high-energy devices prefer to operate large models. Unfortunately, FL can only aggregate the local models under the same architecture (McMahan et al., 2017), so is able to train either small or large models at a time. To cope with heterogeneous energy capacity, one should therefore perform

*Equal contribution :Hankyul Baek and Won Joon Yun.
[1]Korea University, Seoul 02841, Korea. [2]Deakin University, Geelong, VIC 3220, Australia. [3]The University of Utah, UT 84112, USA. [4]University of Oulu, Oulu 90014, Finland. Correspondence to: Jihong Park <jihong.park@deakin.edu.au>, Joongheon Kim <joongheon@korea.ac.kr>.

This work was presented at the International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML 2021 (FL-ICML'21). This workshop does not have official proceedings and this paper is non-archival.

FL two times with the increased overall training time, or run FL simultaneously for two separate groups of devices with reduced training samples while compromising accuracy. Performing FL using a width-controllable *slimmable neural network (SNN)* architecture enables to train these two-level models at once while federating across all devices, after which each trained local SNN model can adjust its width due to its desired energy consumption (Yu & Huang, 2019; Yu et al., 2018).

Heterogeneous Communication Throughput Problem. Unlike wired connectivity, wireless communication channel conditions vary across different devices. When the channel information is known before transmission, poor-channel devices can only exchange small models, while good-channel devices can participate in FL using large models. SNNs allows them to collaborate together, in a way that poor-channel devices send their local SNNs after reducing the widths, and contribute only to a fraction of the entire global model construction. This however entails extra communication and energy costs for probing channel conditions (Ozdemir & Arslan, 2007) that change over time and locations due to random fading and device mobility (Tse & Viswanath, 2005).

Spurred by the aforementioned problems, in this article we propose the first *SNN-based FL algorithm* that leverages *superposition coding (SC)* and *successive decoding (SD)*, coined *slimmable FL (SlimFL)*. By applying SNNs to FL, SlimFL can address the heterogeneous energy capacity problem. Besides, by exploiting SC and SD, SlimFL can proactively cope with the heterogeneous channel throughput problem for unknown channel state information.

To illustrate, consider an SNN with two width levels as shown in Fig. 1. In the uplink from each device to the server, the device uploads its local updates after jointly encoding the left-half (LH) and the right-half (RH) of its local SNN model while allocating different transmission power levels to them, i.e., SC (Cover, 1972). Then, the server first attempts to decode the LH. If decoding the LH is successful, the server successively tries to decode the RH, i.e., SD or also known as successive interference cancellation (SIC) (Vanka et al., 2012). Consequently, when the device-server channel throughput is low, the server can decode only the LH of the uploaded model, obtaining the *half-width (0.5x)* model. When the channel throughput is high, the server can decode both LH and RH, and combine them to yield the *full-width model (1.0x)*. The same principle is also applied in the downlink from the server to each device.

For the two-level SNN architecture under MNIST and Fasion-MNIST classification tasks, simulations results corroborate that SlimFL can simultaneously train both 0.5x and 1.0x models each of which achieves reasonable accuracy and training speed, compared to the vanilla FL baselines having fixed 0.5x and 1.0x widths that should be separately trained without applying SC and SD while incurring 1.5x higher communication costs. What is more, under poor communication channels and non-independent and identically distributed (non-IID) data distributions, SlimFL achieves higher accuracy and lower energy costs (thanks to faster convergence) than its vanilla FL counterpart whose training becomes unstable in the same conditions.

2. Related Work

Depth/Width-Adjustable Neural Networks. To meet different on-device energy and memory requirements, it is common to prune model weights (Han et al., 2016; Seo et al., 2021) or transfer a large trained model's knowledge into a small empty model via knowledge distillation (KD) (Hinton et al., 2015; Seo et al., 2020), which however incurs additional training operations. Alternatively, one can adjust a trained model's width and/or depth in accordance with the resource requirements. Following this principle, depth-controlled neural networks (Kim et al., 2019) and adaptive neural networks (Hu et al., 2019) can adjust their depths after training, whereas SNNs tune their widths (Yu & Huang, 2019; Yu et al., 2018). In this paper, we leverage width-controllable SNNs, and develop its FL version, SlimFL. Such an extension is non-trivial, and entails several design issues, such as local SNN training algorithms, aggregating segment prioritization (e.g., more aggregating the same LH/RH segments vs. balancing the LH and RH aggregations), which will be discussed in Sec 4.

Superposition Coding and Successive Decoding. Since its invention by Thomas Cover in 1972 (Cover, 1972), SC has been widely utilized in communication systems, particularly for simultaneously supporting different devices in the context of non-orthogonal multiple access (NOMA) (Ding et al., 2017). In a nutshell, SC encodes two different data signals into one while allocating two different power levels before transmissions. After receptions, SD decodes the SC-encoded signal by first decoding the stronger signal, followed by subtracting it and decoding the remainder as the weaker signal (Tse & Viswanath, 2005). The same principle is also applicable for supporting a single device simultaneously requesting two types of data with different priorities (Park & Bennis, 2018), such that the higher priority signal should almost surely be decoded while the lower priority signal can be successively decoded only under good channel conditions. Inspired by this, SlimFL makes an SNN's LH a higher priority so as to receive the 0.5x model even under poor channels. It can decode the SNN's RH only when the channel conditions are good, obtaining the 1.0x model by combining both LH and RH. Consequently, SlimFL ensures stable convergence under poor channels, which will be studied in Sec. 5.

3. Preliminaries: SC and SD

Decoding Success Probability. The successful reception of a wireless signal is mainly affected by the signal-to-interference-plus-noise ratio (SINR) (Tse & Viswanath, 2005). At a receiver, the SINR is given as $\gamma = gd^{-\beta}P^T/(\sigma^2 + P^I)$, where P^T, P^I, and σ^2 stand for the transmission power, the received interference power, and the noise power. The d is the transmitter-receiver distance, $\beta \geq 2$ is the path loss exponent, and g is a random small-scale fading gain. Following the Shannon's capacity formula with a Gaussian codebook (Shannon, 1948), the received throughput R with the bandwidth W is given as:

$$R = W \log_2(1+\gamma) \quad \text{(bits/sec)}. \tag{1}$$

When the transmitter encodes raw data with a code rate t, its receiver can successfully decode the encoded data if $R \geq t$. The decoding success probability is derived as:

$$\Pr(R \geq t) = \Pr\left(\frac{gd^{-\beta}P^T}{\sigma^2 + P^I} \geq t'\right), \tag{2}$$

where $t' = 2^{\frac{t}{W}} - 1$.

Superposition Coding (SC). We consider simultaneously conveying K messages from a transmitter to its receiver. These messages are SC-encoded before transmission (Cover, 1972), while the total transmission power budget P^T is allocated to the k-th message with the amount of $P^T = \sum_{k=1}^{K} P_k^T$ transmission power for $k \in [1, K]$. When conveying only a single message, i.e., $K = 1$, there exists no interference at reception, i.e., $P^I = 0$. For $K > 1$, SD determines the interference as elaborated next.

Successive Decoding (SD). At its receiver, the SC-encoded K messages are supposed to be successively reconstructed by first decoding the stronger signal, followed by cancelling out the reconstructed (stronger) signal and then decoding the next stronger signal, i.e., SD, also known as successive interference cancellation (Choi, 2017; Vanka et al., 2012). Assuming $P_k^T > P_{k'}^T$ for $k' > k$, the receiver can successively decode the k-th message while experiencing the rest of the messages as its interference P_k^I, i.e.,

$$P_k^I = gd^{-\beta}\hat{P}_k^I, \tag{3}$$

where $\hat{P}_k^I := \sum_{k'=k+1}^{K} P_{k'}^T$ for $k \leq K-1$, and $\hat{P}_K^I = P_K^I = 0$ as there is no interference for the last message. Let R_k denote the throughput for the k-th message. By substituting (3) into (2), the distribution of R_k is cast as, $\Pr(R_k \geq t) = \Pr\left(g \geq \frac{c}{P_k^T/t' - \hat{P}_k^I}\right)$, where $c = \sigma^2 d^\beta$. Applying this result, the decoding success probability p_k of the k-th message is derived as:

$$\begin{aligned} p_k &= \Pr(R_1 \geq t, R_2 \geq t, \cdots, R_k \geq t) \\ &= \Pr\left(g \geq \max\left\{\frac{c}{P_1^T/t' - \hat{P}_1^I}, \cdots, \frac{c}{P_k^T/t' - \hat{P}_k^I}\right\}\right). \end{aligned} \tag{4}$$

The derivation details of (4) are deferred to Appendix D.

In SlimFL, as Fig. 1 shows, we consider $K = 2$ while treating the LH and RH segments of each model as 2 messages. SC and SD are applied in both uplink (from each device to the server) and downlink (from the server to each device) to be elaborated in Sec. 4. The impact of SC and SD on each uplink or downlink is studied in Appendix G.

4. SlimFL with SC and SD

In this section, we elaborate the architectures and training algorithms of SlimFL. The network under study consists of N devices. Following the federated averaging algorithm (McMahan et al., 2017), the n-th device trains a local SNN model θ^n using its local dataset, and communicates with a parameter server storing a global model. The SNN θ^n is divided into the LH segment $\theta_{1/2}^n$ and the RH segment $\theta_{2/2}^n$, such that the 0.5x model is $\theta_{1/2}^n$ and the 1.0x model is the stack of both segments, i.e., $\theta^n = \mathsf{Stack}\{\theta_{1/2}^n, \theta_{2/2}\}$.

In the uplink, the device uploads the SC-encoded local model θ^n to the server. After reception, according to (4) with $K = 2$ and $P_1^T \gg P_2^T$, the server can successively decode using SD and obtain: (i) the 1.0x model if the channel fading gain satisfies $g \geq \max\{c/(P_1^T/t' - P_2^T), c/(P_2^T/t')\}$; (ii) the 0.5x model if $g \geq c/(P_1^T/t' - P_2^T)$; and (iii) otherwise it obtains no model. In the downlink, SC and SD are also applied, so the global model is obtained at each device according to the aforementioned conditions.

In between the uplink and downlink, the device locally trains its SNN. Existing SNN architectures and training algorithms are intended for standalone learning (Yu & Huang, 2019). Towards improving the scalability for federated learning, we propose a lighter model architecture and training algorithm as detailed next.

Ultra Light MobileNet. The state-of-the-art SNN architecture is the US-MobileNet proposed in (Yu & Huang, 2019). As opposed to a *de facto* standard neural network architecture with a universal batch normalization (BN) layer, US-MobileNet is equipped with multiple separate BN layers to cope with all slimmable model configurations. While effective in standalone learning, in SlimFL with wireless connectivity, not all slimmable model segments are exchanged due to insufficient communication throughput, while the exchanged model segements are aggregated across devices, diluting the effectiveness of BN. In our experiments we even observed training convergence failures

Description	Value
Learning rate (η)	10^{-3}
Optimizer	Adam
Distance (d)	1
Path loss exponent (β)	2.5
Bandwidth per device (W)	1.15×10^5 [Hz]
Uplink transmission power (P_{up}^T)	25 [mW]
Downlink transmission power (P_{dn}^T)	100 [mW]
Noise power (σ_g^2) – good case	-80.6 [dB/Hz]
Noise power (σ_p^2) – poor case	-90.6 [dB/Hz]
UL–MobileNet Layers	**Weight connection of layers 1.0x (0.5x)**
Convolution layer	$C_{in} \times C_{out} \times K \times K$
Conv2D + ReLU6	$1 \times 32 \times 3 \times 3$ ($1 \times 16 \times 3 \times 3$)
Conv2D + ReLU6	$1 \times 32 \times 3 \times 3$ ($1 \times 16 \times 3 \times 3$)
Conv2D + ReLU6	$32 \times 32 \times 1 \times 1$ ($16 \times 16 \times 1 \times 1$)
Conv2D + ReLU6	$1 \times 32 \times 3 \times 3$ ($1 \times 16 \times 3 \times 3$)
Conv2D + ReLU6	$32 \times 64 \times 1 \times 1$ ($16 \times 32 \times 1 \times 1$)
Fully connected layer	$C_{in} \times C_{out}$
Linear	64×10 (32×10)

Table 1. Parameters and model architecture of UL–MobileNet, where C_{in}, C_{out}, K, and ReLU6 stand for dimension of input channel, dimension of output channel, kernel size, and ReLU6$(x) = \min(\max(0, x), 6)$, respectively.

due to BN. Furthermore, managing multiple BN layers not only consumes additional memory costs, but also entails high computing computing overhead. For these reasons, we remove BN layers, and consider a lighter version of US-MobileNet, named *Ultra Light MobileNet (UL-MobileNet)*, with the specifics provided in Tab. 1. Compared to US-MobileNet with more than 100M FLOPS, UL-MobileNet costs only 2.76M FLOPS.

Superposition Training. The sandwich rule and the in-place distillation algorithm are two notable techniques for training an SNN (Yu & Huang, 2019), where the 1.0x model becomes a teacher guiding its sub-width models via knowledge distillation. By nature, it gives more benefits under a larger SNN (i.e., a better teacher) having more sub-width configurations (i.e., more students). This is unfit for our case with the UL-MobileNet having only two width configurations, the 0.5x and 1.0x models. Furthermore, distillation requires to sequentially train the teacher and student models, incurring additional memory and computing costs. Alternatively, inspired a technique for a depth-controllable neural network (Hu et al., 2019), we consider *superposition training* in which the cross-entropy loss $l(\theta_{1/2}^n)$ for the 0.5x model and the loss $l(\theta^n)$ for the 1.0x model are weighted averaged, yielding the following weight update rule:

$$\theta^n \leftarrow \theta^n - \eta \left(w_1 \nabla_{\theta_{1/2}^n} l(\theta_{1/2}^n) + w_2 \nabla_{\theta^n} l(\theta^n) \right), \quad (5)$$

where w_1 and w_2 are positive constants, and η is a learning rate. With such lighter training overhead, in our SlimFL experiments, superposition training achieved even slightly higher accuracy than that with the sandwich rule and in-place distillation, as further discussed in Appendix A. The overall SlimFL operations are summarized in Algorithm 1.

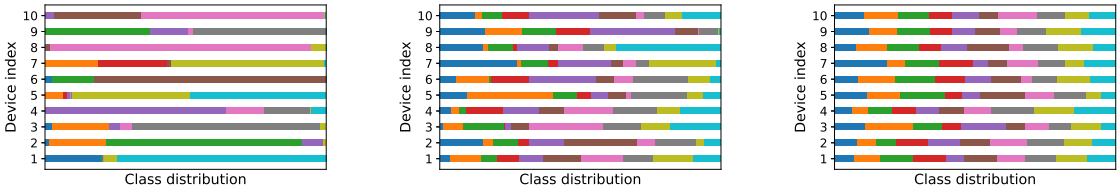
Algorithm 1 SlimFL with SC and SD

Initialize train parameters $\Theta = \{\theta^1, \cdots, \theta^n, \cdots, \theta^N, \theta^G\}$.
Split dataset $\mathcal{D}$ into N datasets $\mathcal{D} = \{D_1, \cdots, D_n, \cdots, D_N\}$.
while *Training* **do**
  LOCAL MODEL TRAINING
  for $n = 1, \cdots, N$ **do**
    for *batch in* D_n **do**
      Update local model parameter θ^n Eq.(5)
    end
  end
  SD-BASED SERVER AGGREGATION
  if *Aggregation Period* **then**
    $C_{1/2} \leftarrow 0, C_{2/2} \leftarrow 0, I_{1/2} \leftarrow \varnothing, I_{2/2} \leftarrow \varnothing$
    for $n = 1, \cdots, N$ **do**
      $g \leftarrow rand(1)$
      if $g \geq p_2^n$ **then**
        $I_{2/2} \leftarrow I_{2/2} \cup n, C_{2/2} \leftarrow C_{2/2} + 1$
      end
      if $p_1^n \leq g < p_2^n$ **then**
        $I_{1/2} \leftarrow I_{1/2} \cup n, C_{1/2} \leftarrow C_{1/2} + 1$
      end
    end
    ▷ **Case1.** $C_{1/2} > 0$, $C_{2/2} > 0$
    $\theta^G \leftarrow \sum_{i \in I_{1/2}} (\theta_{1/2}^i)/C_{1/2} + \sum_{j \in I_{2/2}} (\theta^j)/C_{2/2}$
    ▷ **Case2.** $C_{1/2} > 0$, $C_{2/2} \leq 0$
    $\theta^G \leftarrow \sum_{i \in I_{1/2}} (\theta_{1/2}^i)/C_{1/2}$
    ▷ **Case3.** $C_{1/2} \leq 0$, $C_{2/2} \leq 0$
    pass
  end
  SD-BASED LOCAL UPDATE
  for $n = 1, \cdots, N$ **do**
    $g \leftarrow rand(1)$
    if $g \geq p_2^n$ **then**
      $\theta^n \leftarrow \theta^G$
    end
    if $p_1^n \leq g < p_2^n$ **then**
      $\theta_{1/2}^n \leftarrow \theta_{1/2}^G$
    end
  end
end

5. Evaluation

To show the effectiveness and feasibility of SlimFL, in this section we present the performance of SLimFL exploiting SC and SD compared to its Vanilla FL counterpart without SC nor SD, in terms of accuracy, communication efficiency, and energy efficiency, as well as their robustness to various channel conditions and non-IID data distributions.

Baselines. Our goal is enabling each device to obtain both large and small models so as to cope with its large and small energy levels in future. To this end, by leveraging SNNs with SC and SD, SlimFL simultaneously exchanges and trains 0.5x and 1.0x models by consuming the per-device bandwidth W, uplink transmission power P_{up}^T, and downlink transmission power P_{dn}^T. This is compared with a Vanilla FL baseline, *Vanilla FL*-1.5x. Due to the lack of width-adjustable SNNs, each device in Vanilla FL-1.5x



(a) $\alpha = 0.1$ (extremely non-IID). (b) $\alpha = 1.0$. (c) $\alpha = 10$ ($\approx$ IID).

Figure 2. Data distributions across devices for the different values of the Dirichlet concentration ratio α.

Description		1.0x	0.5x
Computation	MFLOPS / round	2.76	0.79
	# of parameters	4,586	2,293
	Bits / round	172,688	86,344
Transmission Power (P^T)	Uplink w. SC+SD	5.0 [mW]	20.0 [mW]
	Downlink, w. SC+SD	20.0 [mW]	80.0 [mW]

Table 2. Computing costs of UL-MobileNet and transmission power in the uplink and downlink.

Channel Condition	Decoding Success Probability							
	p_1^{up}	p_2^{up}	p_{cp1}^{up}	p_{cp2}^{up}	p_1^{dn}	p_2^{dn}	p_{cp1}^{dn}	p_{cp2}^{dn}
Good	0.983	0.964	0.993	0.973	0.996	0.991	0.998	0.993
Poor	0.810	0.632	0.912	0.704	0.948	0.891	0.977	0.916

Table 3. Decoding success probability under different channel conditions.

separately runs fixed-width 0.5x and 1.0x models, referred to as *Vanilla FL*-0.5*x* and *Vanilla FL*-1.0*x*, respectively. Without SC nor SD, the device exchanges both 0.5x and 1.0x models separately. In brief, Vanilla FL-1.5x is tantamount to simultaneously running the two federated averaging operations separately for 0.5x and 1.0x models by doubling the bandwidth, transmission power, and computing resources. For clarity, we report the performance of Vanilla FL-0.5x and Vanilla FL-1.0x individually if available (i.e., accuracy, received bits), and otherwise we report only Vanilla FL-1.5x (i.e., energy cost).

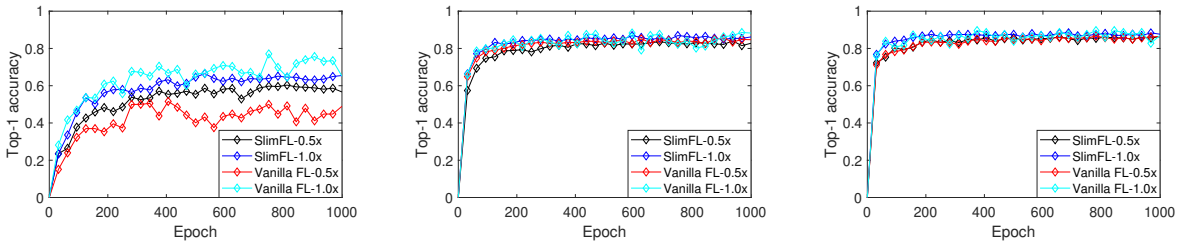
Simulation Settings.

We consider a classification task by default with the Fashion MNIST dataset. Following the method proposed in (Hsu et al., 2019), the non-IIDness of the dataset distribution across devices is controlled by the Dirichlet distribution with its concentration parameter $\alpha \in \{0.1, 1.0, 10\}$, where a lower α is more non-IID distributed (i.e., more imbalanced numbers of samples over labels across devices), as visualized in Fig. 2. For a limited set of simulations, the MNIST dataset is also studied, which is deferred to Appendix C. A single round of uplink and downlink communications is followed by every single local training epoch. The communication channels over different devices are orthogonal in both uplink and downlink. The small-scale fading gain g for each channel realization follows an exponential distribution $g \sim \mathsf{Exp}(1)$, i.e., Rayleigh fading (Tse & Viswanath, 2005). Other communication and training hyperparameters as well as the SNN model architecture are summarized in Tab. 1. For the given transmission power levels and channel environments, the decoding success probabilities in the uplink and downlink are provided in Tab. 3, which is based on (4) and calculated using (15) and (16) in Appendix D.

Robustness to Poor Channels. Fig. 3 and Tab. 4 show that both SlimFL and Vanilla FL achieve high accuracy in good channel conditions. However as the channel condition deteriorates from good to poor channels, Fig. 4 and Tab. 4 illustrate that the maximum accuracy of Vanilla FL-1.0x at $\alpha = 10$ drops from 86% to 82%. Meanwhile, the accuracy of Slim-FL-1.0x keeps the same maximum accuracy 87% at $\alpha = 10$ under both good and poor channels. What is more, at $\alpha = 0.1$, SlimFL-1.0x even achieves 18% higher top-1 accuracy than Vanilla FL-1.0x that consumes more communication and computing costs. Furthermore, the std of Vanilla FL-1.0x's top-1 accuracy increases by up to 59% as channel condition deteriorates, whereas that of SlimFL increases by only up to 31%. These results advocate the robustness of SlimFL against poor channels, as well as its robustness to non-IID data distributions (i.e., low α) and communication efficiency, as elaborated next.

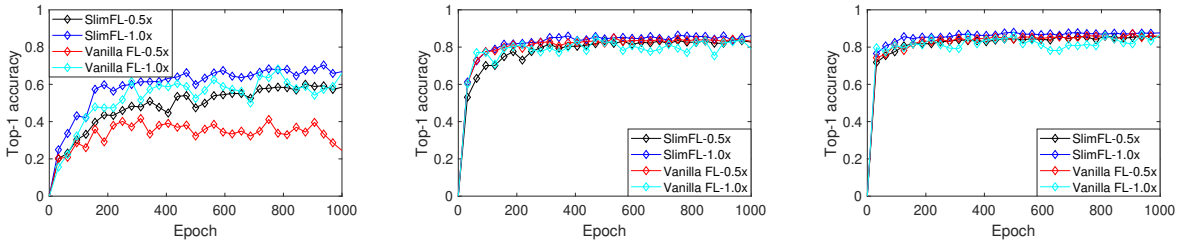
Robustness to Non-IID data. As illustrated in Fig. 4 and Tab. 4, SlimFL-0.5x shows a stable convergence under the conditions of α. Vanilla FL-0.5x and Vanilla FL-1.0x exhibit the std of 8.3 and 9.2, in poor communication condition and with the non-IID dataset ($\alpha = 0.1$). On the contrary, both SlimFL-0.5x and SlimFL-1.0x exhibit the std of 2.4 and 2.9 at top-1 accuracy respectively. This tendency holds even when $\alpha = 1$, $\alpha = 10$. SlimFL-1.0x and SlimFL-0.5x exhibit lower variation than Vanilla FL-1.0x and Vanilla FL-0.5x. This underscores the robustness of SlimFL to non-IID data distributions in poor channels.

Communication Efficiency. The total amount of transmitted bits between 10 devices and server in ideal channel conditions (i.e., always successful decoding) are 205.8MBytes for SlimFL and Vanilla FL-1.0x, and 102.9MBytes for Vanilla FL-0.5x. Tab. 5 shows that SlimFL achieves up to 3.52% less dropped bits than Vanilla FL-1.0x, thanks to the



(a) $\alpha = 0.1$ (b) $\alpha = 1.0$ (c) $\alpha = 10$

Figure 3. Test accuracy in good channel conditions (on average).



(a) $\alpha = 0.1$ (b) $\alpha = 1.0$ (c) $\alpha = 10$

Figure 4. Test accuracy in poor channel conditions (on average).

Method	Top-1 Accuracy (%)					
	Good			Poor		
	$\alpha = 0.1$	$\alpha = 1$	$\alpha = 10$	$\alpha = 0.1$	$\alpha = 1$	$\alpha = 10$
SlimFL-0.5x	54 ± 2.2	83 ± 1.0	85 ± 1.0	56 ± 2.4	82 ± 1.7	85 ± 1.1
SlimFL-1.0x	59 ± 2.3	85 ± 1.1	87 ± 1.1	65 ± 2.9	84 ± 1.4	87 ± 0.9
Vanilla FL-0.5x	45 ± 5.9	84 ± 1.1	85 ± 1.0	39 ± 8.3	83 ± 1.2	85 ± 0.9
Vanilla FL-1.0x	69 ± 5.8	85 ± 4.0	86 ± 4.3	55 ± 9.2	80 ± 6.0	82 ± 4.7

Table 4. Accuracy under different channel conditions and α.

Decoding Success Bits [MBytes]	SlimFL			Vanilla FL-0.5x		Vanilla FL-1.0x	
	0.5x	1.0x	drop	0.5x	drop	1.0x	drop
Uplink (Good)	1.96	198.45	5.46	102.21	0.72	200.30	5.56
Downlink (Good)	0.52	204.01	1.34	102.72	0.21	204.42	1.44
Uplink (Poor)	18.32	130.10	57.44	93.87	9.06	144.93	60.96
Downlink (Poor)	5.87	183.42	16.57	100.56	2.37	188.57	17.29

Table 5. Successfully decoded bits of SlimFL, Vanilla FL-0.5x, and Vanilla FL-1.0x.

use of SC and SD. The reduced dropped bits of SlimFL can be found by the successfully decoded bits of 0.5x models that cannot be simultaneously received under Vanilla FL-1.0x. Note that SlimFL decodes less 1.0x model bits than Vanilla FL-1.0x, as a part of transmission power of SlimFL is allocated to 0.5x models. In return, SlimFL not only receives 1.0x models but also 0.5x models simultaneously. The additionally received 0.5x models correspond to the LH parts of the 1.0x models, which therefore improve the accuracy and convergence speed of both 0.5x and 1.0x models. SlimFL enjoys the aforementioned benefits while consuming only the half of the transmission power and bandwidth compared to Vanilla FL-1.5x, as illustrated in Tab. 5, corroborating its communication efficiency.

Energy Efficiency. Thus far we have measured the performance of SlimFL after training with a fixed 1000 epochs. Here, we measure the energy expenditure until convergence, where the training convergence is defined by the moment when the standard deviation (std) of test accuracy is below a target threshold and the minimum test accuracy becomes higher than the average test accuracy in 100 consecutive rounds (see the details in Appendix E). Given the communication and computing energy costs per round in Tab. 6, Tab. 7 compares the total energy costs of SlimFL and Vanilla FL-1.5x until convergence. The results show that on average, SlimFL achieves 3.6x less total computing cost with 2.9x lower total communication cost until convergence. Such higher energy efficiency comes from the faster convergence of SlimFL even under non-IID distributions and/or poor channel conditions thanks to SC and SD.

Metric	SlimFL	Vanilla FL-1.5x
Communicational Cost [mW/Round]	125	250
Computational Cost [MFLOPS/Epoch]	3.56	3.56

Table 6. Transmission energy and computational cost per communication round.

Metric	non-IIDness	SlimFL		Vanilla FL-1.5x	
		Good	Poor	Good	Poor
Communicational Cost [W]	$\alpha = 0.1$	44.5	35.9	99.5	123.3
	$\alpha = 1.0$	5.3	6.5	9.9	23.0
	$\alpha = 10$	1.9	2.2	6.4	15.9
Computational Cost [GFLOPS]	$\alpha = 0.1$	1.27	1.02	1.88	2.41
	$\alpha = 1.0$	0.15	0.18	0.22	0.51
	$\alpha = 10$	0.05	0.06	0.14	0.35

Table 7. Total computational cost and transmission power of SlimFL and Vanilla FL-1.5x in good/poor channel condition with various non-IIDness ($\alpha = 0.1, 1.0, 10$).

6. Conclusion

FL is a promising solution to enable high-quality on-device learning while protecting user privacy. However, existing FL solutions cannot cope flexibly with different devices having heterogeneous levels of available energy and channel throughput, without significantly compromising communication and energy efficiencies.

To tackle this problem, we propose a novel FL framework that integrates SC and SD communication methods with a width-adjustable SNN architecture. Extensive experiments verify that SlimFL is a communication and energy efficient solution under various communication environments and data distributions. Particularly under poor channel conditions and non-IID data distributions, SlimFL even achieves higher accuracy and faster convergence as well as lower energy expenditure than its vanilla FL counterpart consuming 2x more communication resources. Studying the impact of more adjustable SNN width levels could be an interesting topic for future work. Another interesting direction is to apply SlimFL for multitask learning in which various width configurations correspond to different tasks.

Acknowledgment

This research was funded by Ministry of Health and Welfare (HI19C0842) and also by Academy of Finland6G Flagship (grant no.318927), project SMARTER, projects EU-ICT IntellIoT and EUCHISTERA LearningEdge, Infotech-NOOR and NEGEIN.

References

Choi, J. Joint rate and power allocation for NOMA with statistical CSI. *IEEE Transactions on Communications*, 65(10):4519–4528, October 2017.

Cover, T. Broadcast channels. *IEEE Transactions on Information Theory*, 18(1):2–14, January 1972.

Ding, Z., Liu, Y., Choi, J., Sun, Q., Elkashlan, M., Chih-Lin, I., and Poor, H. V. Application of non-orthogonal multiple access in LTE and 5G networks. *IEEE Communications Magazine*, 55(2):185–191, February 2017.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *Proceedings of International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, May 2016.

Hard, A., Rao, K., Mathews, R., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C., and Ramage, D. Federated learning for mobile keyboard prediction. *Arxiv preprint*, abs/1811.03604, November 2018.

Hinton, G. E., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. In *Proceedings of Neural Information Processing Systems (Deep Learning and Representation Learning Workshop)*, pp. 1–9, Montréal, Canada, December 2015.

Hsu, T. H., Qi, H., and Brown, M. Measuring the effects of non-identical data distribution for federated visual classification. *ArXiv preprint*, abs/1909.06335, September 2019.

Hu, H., Dey, D., Hebert, M., and Bagnell, J. A. Learning anytime predictions in neural networks via adaptive loss balancing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3812–3821, HI, USA, January 2019.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., et al. Advances and open problems in federated learning. *arXiv preprint*, abs:1912.04977, December 2019.

Kim, D., Kim, J., Kwon, J., and Kim, T.-H. Depth-controllable very deep super-resolution network. In *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, Budapest, Hungary, July 2019.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of Artificial Intelligence and Statistics (AISTATS)*, FL, USA, April 2017.

Ozdemir, M. K. and Arslan, H. Channel estimation for wireless OFDM systems. *IEEE Communications Surveys Tutorials*, 9(2):18–48, 2007.

Park, J. and Bennis, M. URLLC-eMBB slicing to support VR multimodal perceptions over wireless cellular systems. *ArXiv preprint*, abs/1805.00142, May 2018.

Park, J., Samarakoon, S., Elgabli, A., Kim, J., Bennis, M., Kim, S., and Debbah, M. Communication-efficient and distributed learning over wireless networks: Principles and applications. *Proceedings of the IEEE*, 109(5):796–819, May 2021.

Samarakoon, S., Bennis, M., Saad, W., and Debbah, M. Federated learning for ultra-reliable low-latency V2V communications. In *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, Abu Dhabi, United Arab Emirates, December 2018.

Seo, H., Park, J., Oh, S., Bennis, M., and Kim, S.-L. Federated knowledge distillation. *ArXiv preprint*, abs/2011.02367, November 2020.

Seo, S., Ko, S.-W., Park, J., Kim, S.-L., and Bennis, M. Communication-efficient and personalized federated lottery ticket learning. *ArXiv preprint*, abs/2104.12501, April 2021.

Shannon, C. E. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

Shiri, H., Park, J., and Bennis, M. Communication-efficient massive UAV online path control: Federated learning meets mean-field game theory. *IEEE Transactions on Communications*, 68(11):6840–6857, 2020.

Tse, D. N. C. and Viswanath, P. *Fundamentals of Wireless Communications*. Cambridge University Press, 2005.

Vanka, S., Srinivasa, S., Gong, Z., Vizi, P., Stamatiou, K., and Haenggi, M. Superposition coding strategies: Design and experimental evaluation. *IEEE Transactions on Wireless Communications*, 11(7):2628–2639, November 2012.

Yu, J. and Huang, T. S. Universally slimmable networks and improved training techniques. In *Proceedings of International Conference on Computer Vision (ICCV)*, pp. 1803–1811, Seoul, S. Korea, October 2019.

Yu, J., Yang, L., Xu, N., Yang, J., and Huang, T. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, December 2018.

Algorithm 2 USTrain (Yu & Huang, 2019)

Define *width range*, for example, $[0.25, 0.5, 0.75, 1.0]$x.
Define n as number of sampled widths per training iteration, for example, $n = 4$.
Initialize training settings of shared network M.
for $(t = 1, ..., T_{iters})$ **do**
  Get next mini-batch of data x and label y).
  Clear gradients, $optimizer.zero\_grad()$.
  Execute full-network, $y' = M(x)$.
  Compute loss, $loss = criterion(y', y)$.
  Accumulate gradients, $loss.backward()$.
  Stop gradients of y' as label, $y' = y'.detach()$.
  Randomly sample $(n-2)$ widths, as *width samples*.
  Add smallest width to *width samples*.
  for *width in width samples* **do**
    Execute sub-network at *width*, $\hat{y} = M'(x)$.
    Compute loss, $loss = criterion(\hat{y}, y')$.
    Accumulate gradients, $loss.backward()$.
  end
  Update weights, $optimizer.step()$.
end

Appendices

A. Local SNN Training Algorithm

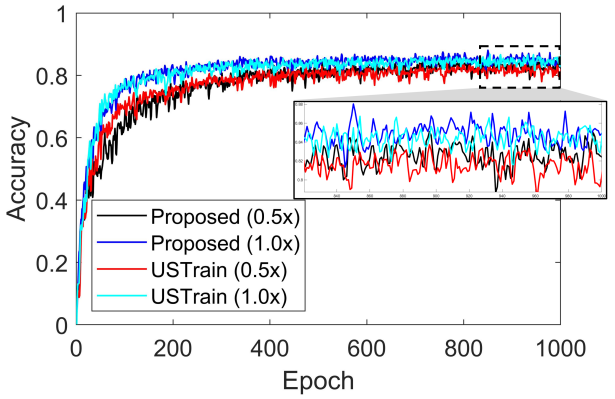


Figure 5. Comparison of two different training methods in poor channel conditions with $\alpha = 1$

The state-of-the-art SNN training algorithm is the USTrain method in Algorithm 2 proposed by (Yu & Huang, 2019). At its core, USTrain utilizes the inplace distillation and the sandwich rule that are effective under the case where an SNN can be divided into more than two segments. The SNN architecture under study consists of only the LH and RH segments, making the Algorithm 2 unfit for our case. Alternatively, we propose a new weighted-sum based SNN training algorithm in (5), which is inspired by the training algorithm used in depth-controllable model training (Hu et al., 2019). Fig. 5 shows our proposed local train-

ing method outperforms USTrain when training an UL-MobileNet SNN model.

B. Optimizer

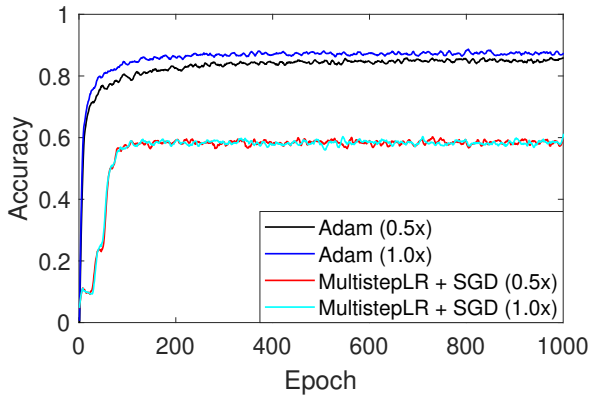


Figure 6. Multi-step-LR + SGD Optimizer in poor channel condition with $\alpha = 1$

The optimizer of universally slimmable neural network consists of stochastic gradient descent optimizer with multi-step learning rate scheduler (Yu & Huang, 2019). However, UL-MobileNet converges to local minima when using multi-step-LR + SGD optimizer as shown in Fig. 6. Therefore, we adopt Adam optimizer as an optimizer of UL-MobileNet.

C. Dataset

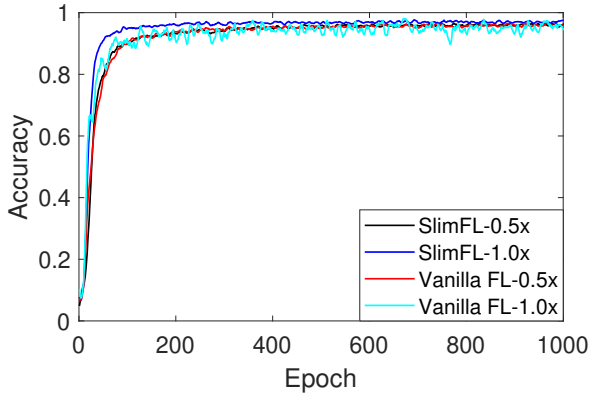


Figure 7. Performance of proposed algorithm and two comparisons with MNIST dataset ($\alpha = 1$) in poor channel condition.

In this paper, as Fig. 7 shows, all models exhibit high accuracy and reasonable convergence when using MNIST dataset. Therefore we use Fashion MNIST dataset to show the differences between models in the main experiment. The result of accuracy is too high as shown in Fig. 7. With MNIST dataset, the performance difference between 0.5x and 1.0 is too trivial so we adopt Fashion MNIST dataset for UL-MobileNet.

D. Decoding Success Probability Derivation

The distribution of the throughput R_k for the k-th message is cast as

$$\Pr(R_k \geq t) = \Pr\left(W \log_2\left(1 + \frac{g d^{-\beta} P_k^T}{\sigma^2 + P_k^I}\right) \geq t\right) \quad (6)$$

$$= \Pr\left(\frac{g d^{-\beta} P_k^T}{\sigma^2 + P_k^I} \geq t'\right), \quad (7)$$

where $t' = 2^{t/W} - 1$. Here, $P_k^I = g d^{-\beta} \hat{P}_k^I$, with $\hat{P}_k^I := \sum_{k'=k+1}^{K} P_{k'}^T$ for $k \leq K-1$, and $\hat{P}_K^I = P_K^I = 0$. By applying this, (7) is recast as

$$(7) = \Pr\left(g d^{-\beta} P_k^T \geq t'\left(\sigma^2 + g d^{-\beta} \hat{P}_k^I\right)\right) \quad (8)$$

$$= \Pr\left(g \geq \frac{c}{P_k^T/t' - \hat{P}_k^I}\right), \quad (9)$$

where $c = \sigma^2 d^{\beta}$. Applying this result, the decoding success probability p_k of the k-th message is represented as

$$p_k = \Pr(R_1 \geq t, R_2 \geq t, \cdots, R_k \geq t) \quad (10)$$

$$= \Pr\left(g \geq \frac{c}{P_1^T/t' - \hat{P}_1^I}, \cdots, g \geq \frac{c}{P_k^T/t' - \hat{P}_k^I}\right) \quad (11)$$

$$= \Pr\left(g \geq \max\left\{\frac{c}{P_1^T/t' - \hat{P}_1^I}, \cdots, \frac{c}{P_k^T/t' - \hat{P}_k^I}\right\}\right). \quad (12)$$

For $K = 2$, $\hat{P}_1^I = P_2^T$ and $\hat{P}_2^I = 0$. Following (12) for $P_1^T \gg P_2^T$ with SD, the decoding success probabilities of the two messages are given as follows:

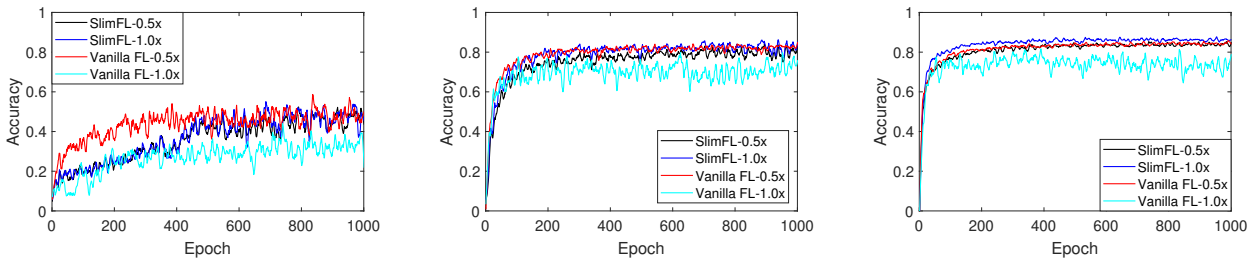
$$p_1 = \Pr\left(g \geq \frac{c}{P_1^T/t' - P_2^T}\right) \quad (13)$$

$$P_2 = \Pr\left(g \geq \frac{c}{P_2^T/t'}\right) \quad (14)$$

Under Rayleigh fading, the small-scale fading power gain g follows an exponential distribution, i.e., $g \sim \mathsf{Exp}(1)$. Applying the complementary cumulative distribution function (CCDF) of an exponential random variable to (13) and (14), the decoding success probabilities finally become:

$$p_1 = \exp\left(-\frac{c}{P_1^T/t' - P_2^T}\right) \quad (15)$$

$$p_2 = \exp\left(-\frac{c}{P_2^T/t'}\right). \quad (16)$$



(a) $\alpha = 0.1$ (b) $\alpha = 1.0$ (c) $\alpha = 10$

Figure 8. Performance difference in extremely poor channel conditions.

E. Convergence Measurement

We design a method to measure the convergence of each model in all the experimental environments presented in Tab. 7. Tab. 4 represents that SlimFL and Vanilla FLs (i.e., 0.5x and 1.0x) have huge differences in std and mean of Top-1 accuracy in identical environments. These differences between SlimFL and Vanilla FLs make it difficult to set common criteria for measuring convergence as shown in Tab. 4. Therefore, to measure the convergence of models, we define the reference values of the mean ($mean_{ref}$) and std (std_{ref}), respectively. To describe in details, we set $mean_{ref}$ as 80% of the last 100 epoch's average Top-1 accuracy and std_{ref} as 7.2%. We define the convergence when average of Top-1 accuracy for 100 consecutive epochs is higher than $mean_{ref}$, and the average std is lower than std_{ref}.

F. Extremely Poor Communication Channels

Condition	Decoding Success Probability							
	p_1^{up}	p_2^{up}	p_{cp1}^{up}	p_{cp2}^{up}	p_1^{dn}	p_2^{dn}	p_{cp1}^{dn}	p_{cp2}^{dn}
Extremely Poor	0.22	0.08	0.50	0.07	0.80	0.64	0.86	0.56
Uplink SC+SD/Downlink 1.0x	0.76	0.70	0.92	0.38	-	0.74	0.97	0.74
Uplink SC+SD/Downlink 0.5x	0.76	0.70	0.92	0.38	0.97	-	0.97	0.74
Uplink 0.5x/Downlink SC+SD	-	0.38	0.92	0.38	0.97	0.94	0.97	0.74

Table 8. Decoding success probabilities under an extremely poor channel condition and transmission method.

We simulate the performance of our proposed model in extremely poor channel conditions as shown in Tab. 8. Fig. 8 presents the learning curve in respect to accuracy of three algorithms (i.e., Proposed, Comp1, and Comp2). With non-IID dataset ($\alpha = 0.1$), Fig. 8(a) shows that the accuracy of SlimFL (both 0.5x and 1x) converges to 0.44 at 620 epochs. Also Comp1 and Comp2 converge to 0.44 and 0.38 at 380 epochs and 910 epochs, respectively. As shown in Fig. 8(b), the accuracy of SlimFL and Comp1 converge to 0.81 and Comp2 converges to 0.76. Finally, using dataset close to IID (i.e., $\alpha = 10$), SlimFL 1.0x shows the highest accuracy, SlimFL 0.5x and Comp1 follow next, and Comp2 shows the lowest accuracy. SlimFL outperforms Comp1 and Comp2 when $\alpha = 1$ and $\alpha = 0.5$ regarding accuracy even in extremely poor channel condition.

G. SC and SD in the Uplink and/or Downlink

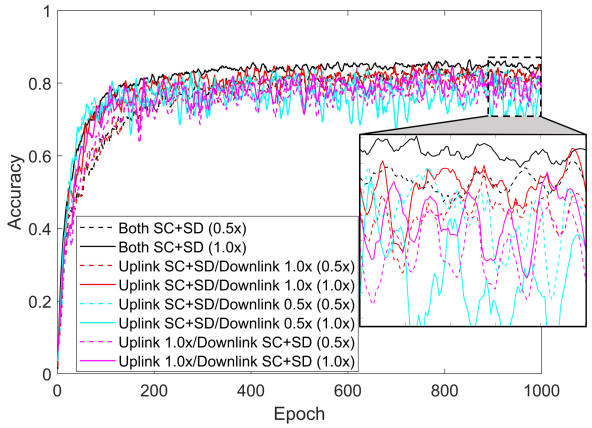


Figure 9. Performance difference in various transmission method (Poor, $\alpha = 1$).

We design our proposed algorithm with utilizing SC and SD in both uplink and downlink communication. We simulated with various transmission methods (e.g. Uplink SC+SD/Downlink 1.0x, Uplink SC+SD/Downlink 0.5x, Uplink 1.0x/Downlink SC+SD). The first comparison (Uplink SC+SD/Downlink 1.0x) is that devices upload both $\theta_{1/2}$ and $\theta_{2/2}$ with SC and SD, and the server transmits 1.0x to devices. The second comparison (Uplink SC+SD/Downlink 0.5x) is that devices upload both $\theta_{1/2}$ and $\theta_{2/2}$ and the server transmits $\theta_{1/2}$ to devices. The third comparison (Uplink 1.0x/Downlink SC+SD) is that devices can transmit only θ and the server can transmit both $\theta_{1/2}$ and $\theta_{2/2}$ to local model. Since Uplink 0.5x/Downlink SC+SD is not proper transmission technique, we do not consider Uplink 0.5x/Downlink SC+SD. The decoding success probability of three techniques are shown in Tab. 8.

We simulated with those three comparison technique, and Fig. 9 shows the result. In Fig. 9, learning curves converge with a similar tendency. At the end of training phase (i.e.,

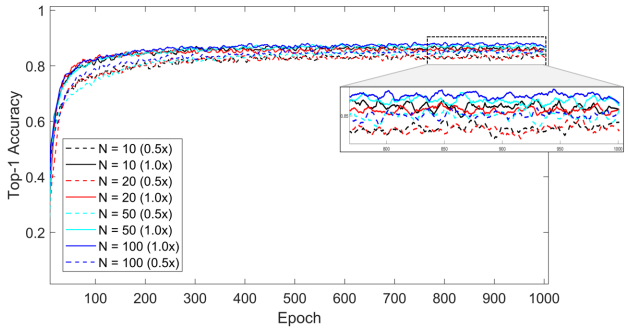


Figure 10. Top-1 accuracy with different number of devices.

800 – 1000 epochs), the accuracy is found to be high in the order of our proposed method (Both SC+SD), Uplink SC+SD/Downlink 1.0x, Uplink 1.0x/Downlink SC+SD, and Uplink 0.5x/Downlink SC+SD. Consequently, our proposed wireless communication system (i.e., both SC+SD) shows the best performance among those comparisons.

H. Scability

Fig. 10 illustrates the top-1 accuracy of SlimFL for a different number N of devices. As N grows from 10 to 100, the accuracy of the 0.5x model increases from 84% to 86% that coincides with the accuracy of the 1.0x model when $N = 10$. In other words, federating with 10x more devices is equivalent to running a 2x larger model. For the 1.0x model, the gain from federating more devices than $N = 10$ is negligible. Studying such a scalability trened under different datasets and/or model architectures would be an interesting topic for future work.