ProgFed: Effective, Communication, and Computation Efficient Federated Learning by Progressive Training

CISPA Helmholtz Center for Information Security, Germany

Hui-Po Wang

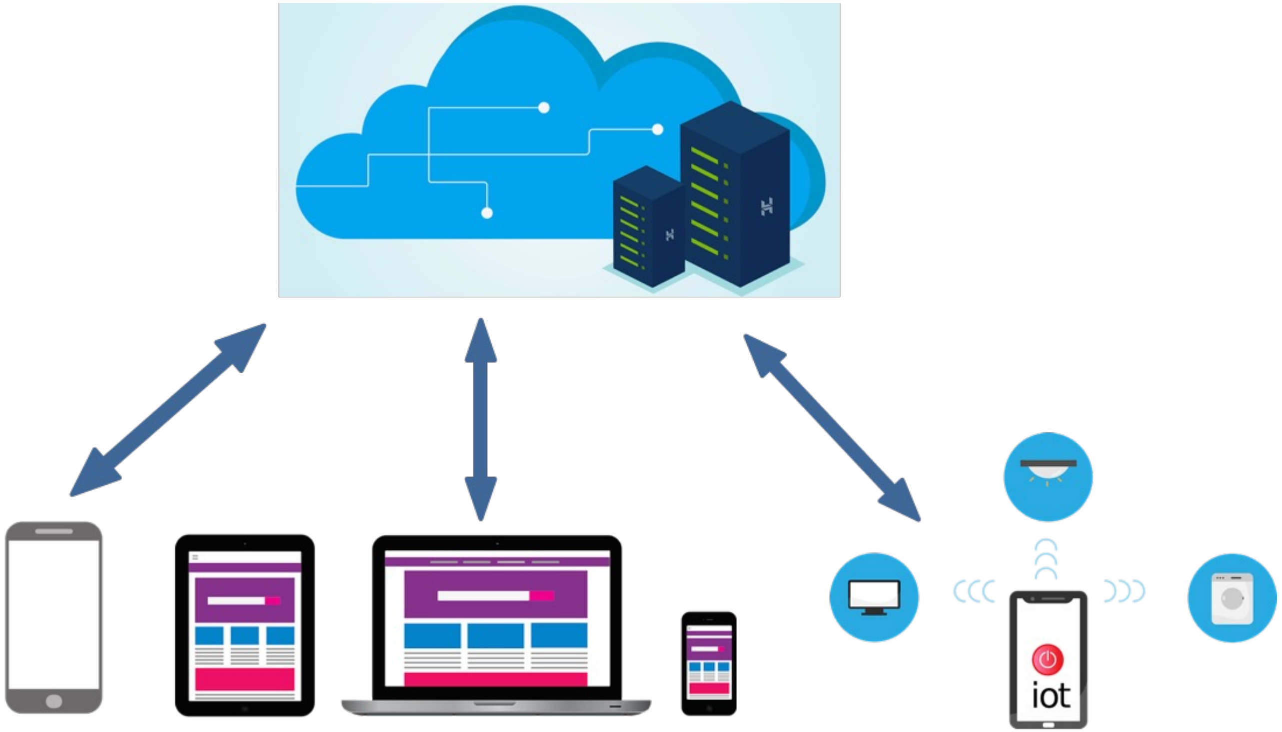
Sebastian U. Stich

Yang He

Mario Fritz

Introduction

- Federated learning advanced applications of large-scale machine learning systems
- **Limited bandwidth** and **computation power** have become the main bottleneck
- How to further reduce the computation and communication costs while retaining utility?



Prior Work

- Message compression includes using fewer bits (i.e., quantization) and only sending partial updates (i.e., sparsification)
- Model pruning identifies a slim network within the original network while retaining performance (usually happens after training)
- Model distillation communicates logits rather than gradients (often requires additional data)
- **In this work, we take advantage of the learning dynamic to reduce the training costs**

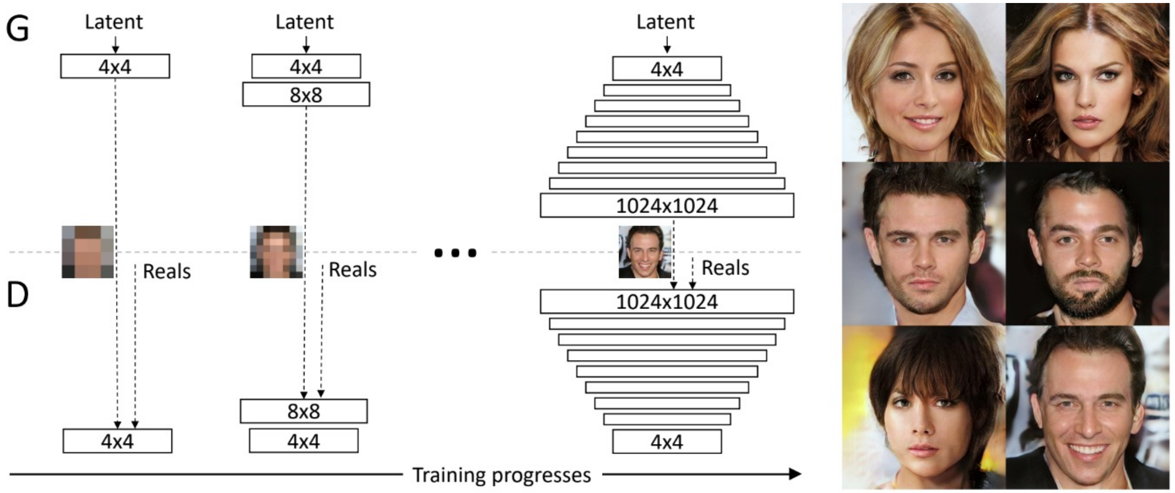
Technique	Computation Reduction	Communication Reduction	Dataset Efficiency
Message Compression	✗	✓	✓
Model Pruning	✓(only for inference)	✗	✓
Model Distillation	✓	✓	✗
ProgFed (Ours)	✓	✓	✓

Progressive Learning and its Challenges in Federated Learning

- In progressive learning, models learn from easier tasks (e.g., lower image resolution) and gradually to complicated tasks (e.g., higher image resolution)

- The growing process inherently reduces the **communication** and **computation** costs

- Challenges
  - Not designed for prediction tasks
  - Not designed for federated learning



Karras et al.

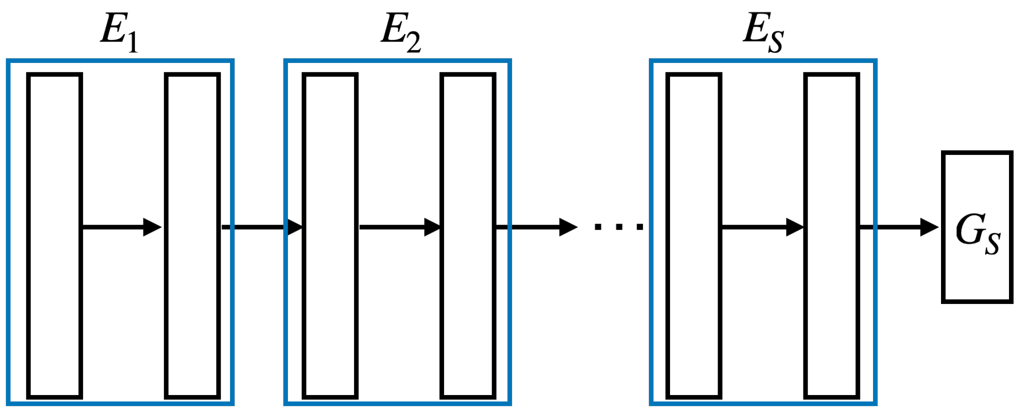
ProgFed

- We propose ProgFed, the first progressive learning framework for federated learning
- We divide the entire model into several disjoint components and introduce temporal heads

$$\mathcal{M} := G_S \circ \bigcirc_{i=1}^{S} E_i = G_S \circ E_S \circ \cdots \circ E_2 \circ E_1 .$$

- Extend progressive learning to federated learning

$$\mathcal{M}^s := G_s \circ \bigcirc_{i=1}^{s} E_i \qquad f^s(\mathbf{x}^s) := \mathcal{L} \circ \mathcal{M}^s(\mathbf{x}^s)$$



Full model

ProgFed

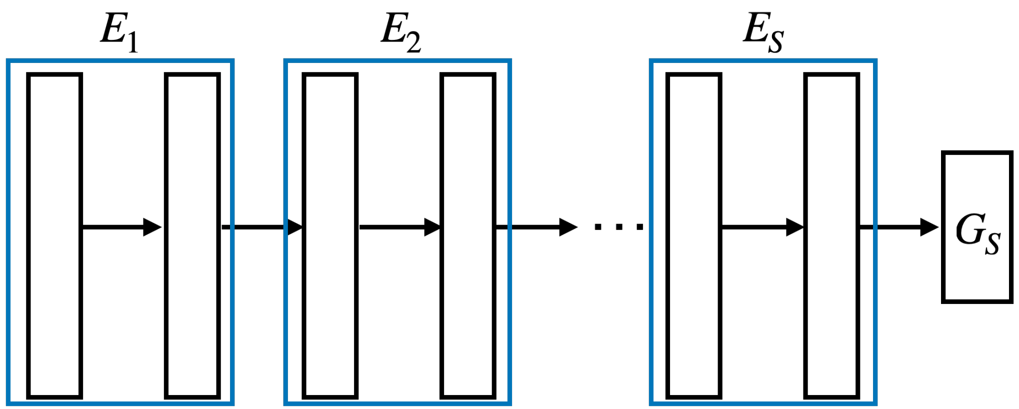
- We propose ProgFed, the first progressive learning framework for federated learning
- We divide the entire model into several disjoint components and introduce temporal heads

$$\mathcal{M} := G_S \circ \bigcirc_{i=1}^{S} E_i = G_S \circ E_S \circ \cdots \circ E_2 \circ E_1 \,.$$

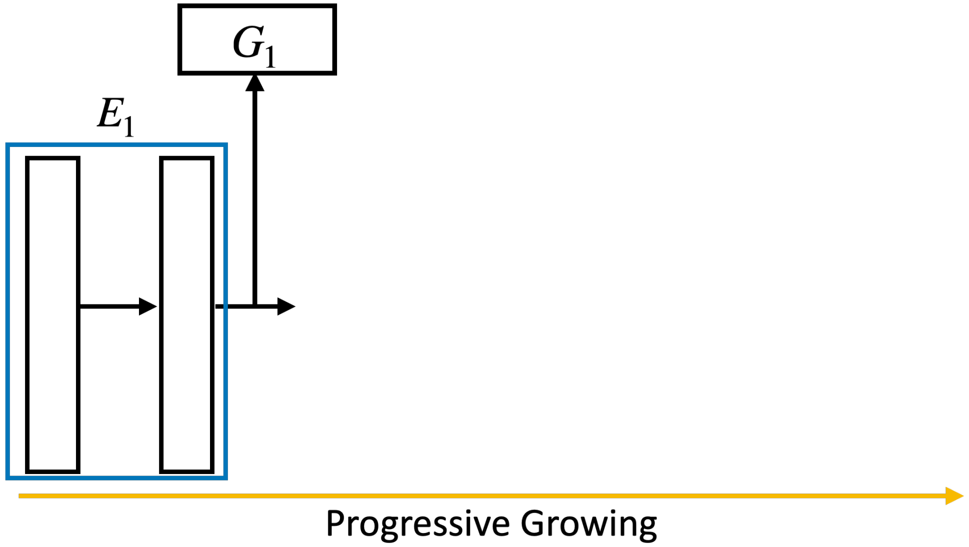
- Extend progressive learning to federated learning

$$\mathcal{M}^s := G_s \circ \bigcirc_{i=1}^{s} E_i$$

$$f^s(\mathbf{x}^s) := \mathcal{L} \circ \mathcal{M}^s(\mathbf{x}^s)$$



Full model



ProgFed

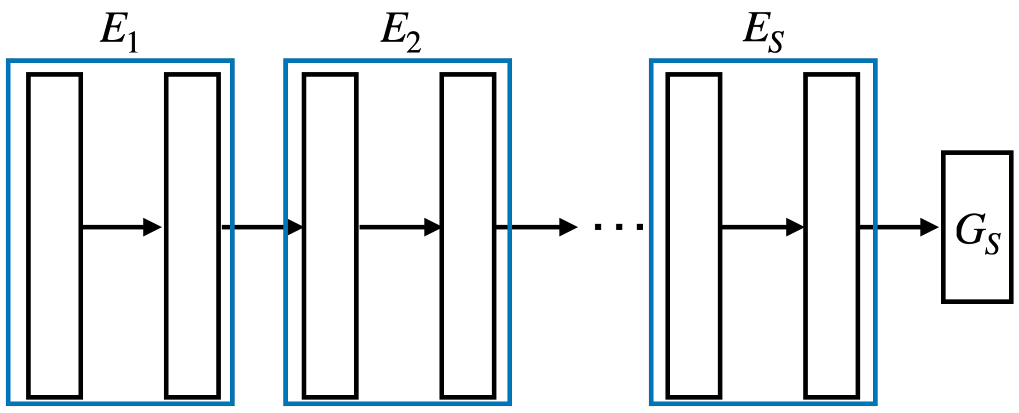
- We propose ProgFed, the first progressive learning framework for federated learning
- We divide the entire model into several disjoint components and introduce temporal heads

$$\mathcal{M} := G_S \circ \bigcirc_{i=1}^{S} E_i = G_S \circ E_S \circ \cdots \circ E_2 \circ E_1 \,.$$

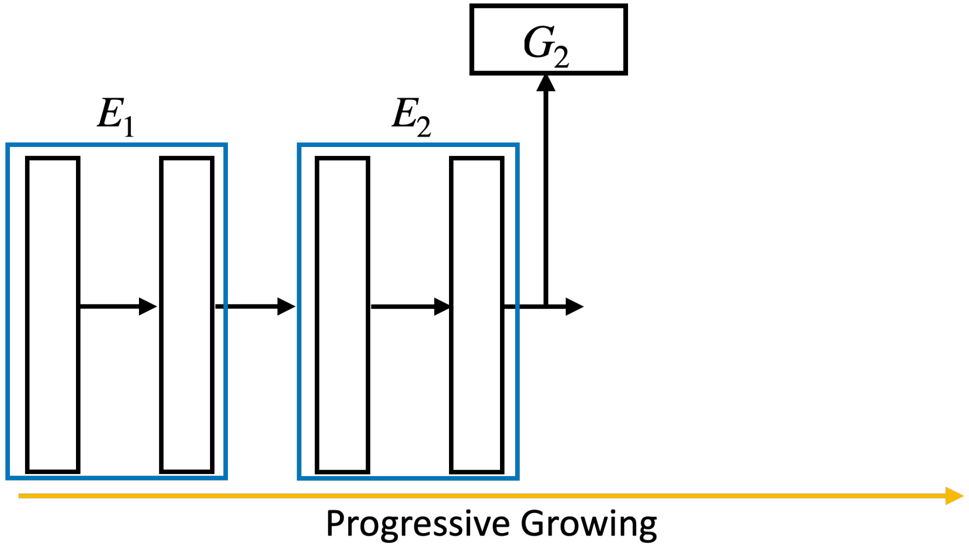
- Extend progressive learning to federated learning

$$\mathcal{M}^s := G_s \circ \bigcirc_{i=1}^{s} E_i$$

$$f^s(\mathbf{x}^s) := \mathcal{L} \circ \mathcal{M}^s(\mathbf{x}^s)$$



Full model



ProgFed

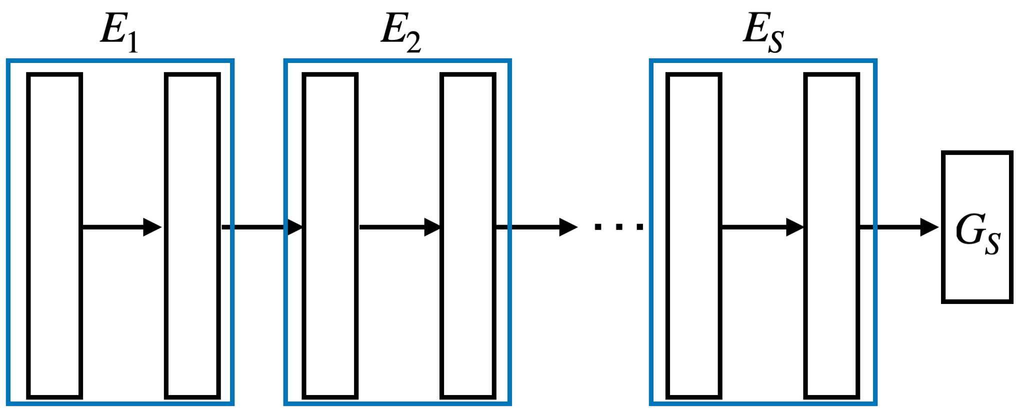
- We propose ProgFed, the first progressive learning framework for federated learning
- We divide the entire model into several disjoint components and introduce temporal heads

$$\mathcal{M} := G_S \circ \bigodot_{i=1}^{S} E_i = G_S \circ E_S \circ \cdots \circ E_2 \circ E_1 .$$

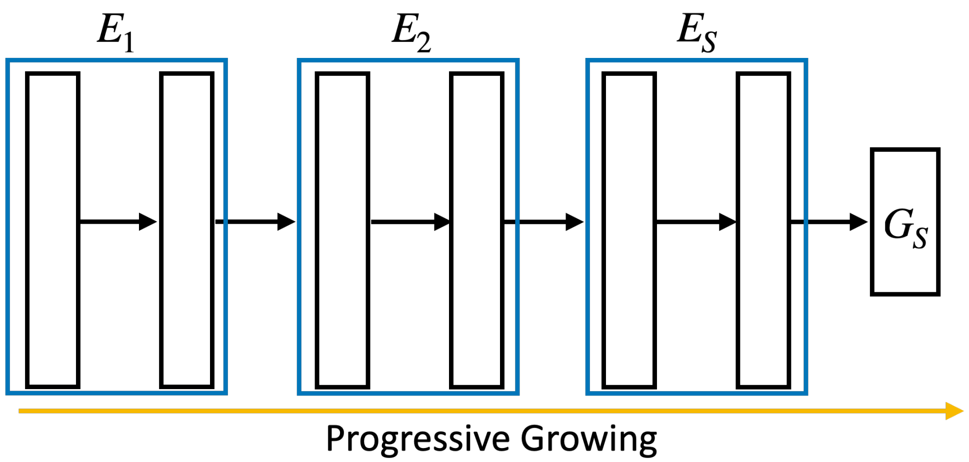
- Extend progressive learning to federated learning

$$\mathcal{M}^s := G_s \circ \bigodot_{i=1}^{s} E_i$$

$$f^s(\mathbf{x}^s) := \mathcal{L} \circ \mathcal{M}^s(\mathbf{x}^s)$$



Full model

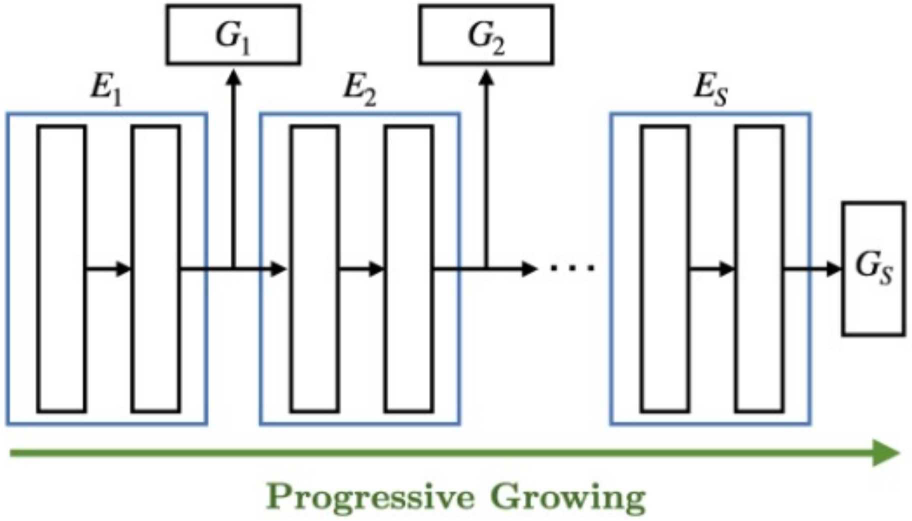


Practical Considerations in ProgFed

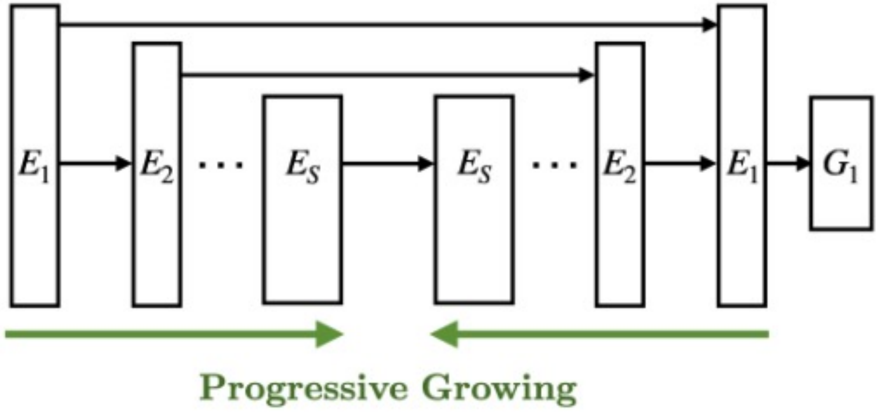
- How do we split the model, and when do we extend the model?
- **Practical Guideline**: the growing cycle (T_s) is controlled by #epochs (T) and #stages (S)

$$T_s = \frac{T}{2S} \text{ for } s < S,\ T_S = \frac{2T(S+1)}{2S}, \text{ such that } T = \sum_{s=1}^{S} T_s$$

- The guideline ensures that we only conduct progressive learning in the first half of training and resume end-to-end training in the rest



(a) Feed-forward networks



(b) U-nets (symmetric growing).

Theoretical Analysis

- We assume the loss functions are *L-smooth* and gradient noise from clients is *bounded* (c.f. Assumption 1 and 2 in our paper)

- Theorem 1 suggests that sub-models converge, and the full model converges at most two times slower than the standard way but with much cheaper per-iteration costs

Theorem 1. Let Assumptions 1 and 2 hold, and let the stepsize in iteration t be $\gamma_t = \alpha_t \gamma$ with $\gamma = \min\left\{\frac{1}{L}, (\frac{F_0}{\sigma^2 T})^{\frac{1}{2}}\right\}$, $\alpha_t = \min\left\{1, \frac{\langle \nabla f(\mathbf{x}_t)_{|E_s}, \nabla f^s(\mathbf{x}_t^s)_{|E_s}\rangle}{\|\nabla f^s(\mathbf{x}_t^s)_{|E_s}\|^2}\right\}$. Then it holds for any $\epsilon > 0$,

- $\frac{1}{T}\sum_{t=0}^{T-1} \alpha_t^2 \left\|\nabla f^s(\mathbf{x}_t^s)_{|E_s}\right\|^2 < \epsilon$, after at most the following number of iterations T:

$$\mathcal{O}\left(\frac{\sigma^2}{\epsilon^2} + \frac{1}{\epsilon}\right) \cdot LF_0\,. \tag{5}$$

- Let $q := \max_{t\in[T]}\left(q_t := \frac{\|\nabla f(\mathbf{x}_t)\|}{\alpha_t \|\nabla f^s(\mathbf{x}_t^s)_{|E_s}\|}\right)$, then $\frac{1}{T}\sum_{t=0}^{T-1} \left\|\nabla f(\mathbf{x}_t)\right\|^2 < \epsilon$ after at most the following iterations T:

$$\mathcal{O}\left(\frac{q^4\sigma^2}{\epsilon^2} + \frac{q^2}{\epsilon}\right) \cdot LF_0\,, \tag{6}$$

where $F_0 := f(\mathbf{x}_0) - (\min_{\mathbf{x}} f(\mathbf{x}))$.

Experiment Settings

- Dataset: EMNIST, CIFAR-10, CIFAR-100, and BraTS
- Centralized settings: ResNet-18, ResNet-152, VGG16, and VGG19 for CIFAR-100
- Federated settings: small ConvNets for EMNIST (3400 clients, non-IID) and CIFAR-10 (100 clients, IID), ResNet-18 for CIFAR-100 (500 clients, non-IID), and U-nets for BraTS (10 clients, IID)
- More details can be found in the paper

Experiments – Computation Efficiency

- We conduct experiments on four architectures and CIFAR-100 in the **centralized** setting

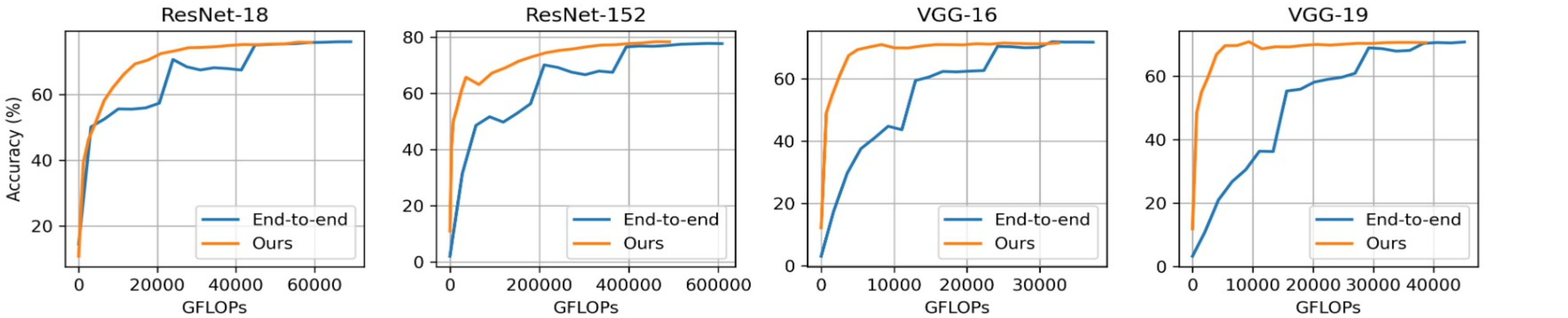


Figure 2: Accuracy (%) vs. GFLOPs on CIFAR-100 in the centralized setting.

Table 2. Results on CIFAR-100 in the centralized setting.

	Accuracy		Reduction	
	End-to-end	Ours	Walltime	FLOPs
ResNet18	**76.08±0.12**	75.84±0.28	-24.75%	-14.60%
ResNet152	77.77±0.38	**78.57±0.33**	-22.75%	-19.68%
VGG16	**71.79±0.15**	71.54±0.45	-14.57%	-13.02%
VGG19	70.81±1.18	**70.90±0.43**	-22.10%	-14.43%

Experiments – Communication Efficiency

- We conduct experiments on **federated** classification and segmentation across various datasets and architectures

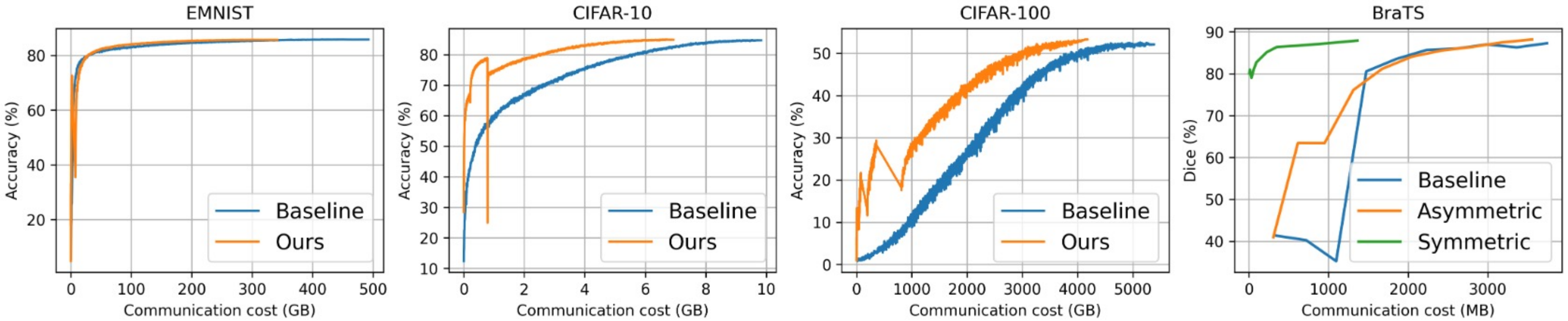


Figure 5: Communication cost vs. Accuracy (%) in federated settings on EMNIST (3400 clients, non-IID), CIFAR-10 (100 clients, IID), CIFAR-100 (500 clients, non-IID), and BraTS (10 clients, IID).

Table 3. Results in federated settings. We report accuracy (%) for classification and Dice scores (%) for segmentation, followed by cost reduction (CR) as compared to the baselines (end-to-end).

	Baseline	Ours	CR
EMNIST	**85.75 ± 0.11**	85.67 ± 0.06	-29.49%
CIFAR-10	84.67 ± 0.14	**84.85 ± 0.30**	-29.70%
CIFAR-100	52.08 ± 0.44	**53.23 ± 0.09**	-22.90%
BraTS (Aym.)	86.77 ± 0.45	**87.66 ± 0.49**	-5.02%
BraTS (Sym.)	86.77 ± 0.45	**87.96 ± 0.03**	-63.60 %

Experiments – Compatibility

Federated ResNet-18 on CIFAR-100 w/ linear quantization (LQ-X) and sparsification (SQ-X)

	Float	LQ-8	LQ-4	LQ-2	SP-25	SP-10	LQ-8 +SP-25	LQ-8 +SP-10
				Accuracy				
Baseline	52.54	49.40	49.55	47.26	51.23	51.79	50.79	50.97
Ours	**53.25**	**53.07**	**52.32**	**52.87**	**52.13**	**51.86**	**52.05**	**52.32**
				Compression Ratio (%)				
Baseline	100	25.00	12.50	6.25	25.00	10.00	6.25	2.50
Ours	**77.10**	**19.28**	**9.64**	**4.82**	**19.28**	**7.71**	**4.82**	**1.93**

Results of ProgFed with FedAvg, FedProx, and FedAdam on CIFAR-100

	EMNIST		
	FedAvg	FedProx	FedAdam
End-to-end	**85.75**	**86.36**	**86.53**
FedProg (S=4)	85.67	86.08	86.13
	CIFAR-100		
	FedAvg	FedProx	FedAdam
End-to-end	52.08	53.25	56.21
FedProg (S=4)	**53.23**	**54.28**	**60.55**

CISPA
HELMHOLTZ CENTER FOR
INFORMATION SECURITY

Thank you for your attention

Our code is available: https://github.com/a514514772/ProgFed