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

Hui-Po Wang  Sebastian Stich  Yang He  Mario Fritz

CISPA Helmholtz Center for Information Security, Germany

## Overview

We find that leveraging learning dynamics helps reduce resource demands in federated learning and propose ProgFed, the first federated progressive learning framework to utilize such a feature. Our method extends progressive learning to general federated prediction tasks and presents efficiency and preferable performance in various settings.

## Contributions

- **Novelty.** We propose ProgFed, the first federated progressive learning framework to reduce the training resource demands
- **Efficiency.** Our method reduces communication and computation costs on various datasets, tasks, and architectures, including 25% computation and up to 32% two-way communication costs in federated classification and 63% in federated segmentation, without sacrificing performance
- **Compatibility.** Our method is compatible with existing compression techniques with up to 5% improvement and advanced federated optimization with up 4.3% improvement

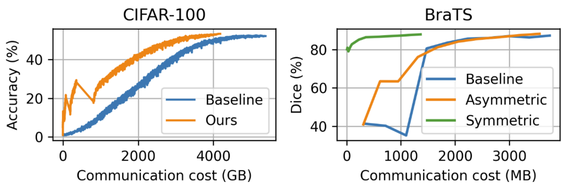



Figure 1:Performance vs. costs in federated settings.

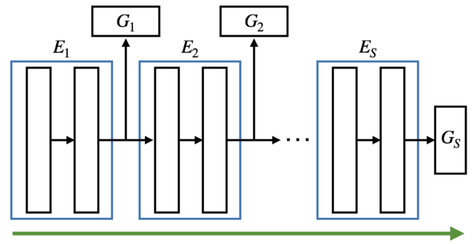
## Project Page

More results and code can found in our project page `https://a514514772.github.io/ProgFed/`.

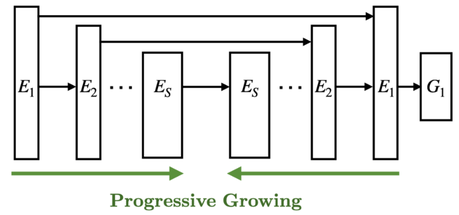
## ProgFed

- **Progressive learning for Federated Prediction Tasks**

Progressive Learning (PL) inherently reduces communication and computation costs, but it is typically designed for *image synthesis in centralized settings*. ProgFed (1) divides the original models into several disjoint components to retain the benefits of PL and (2) introduces *light-weight temporal* classifiers to facilitate training across clients.



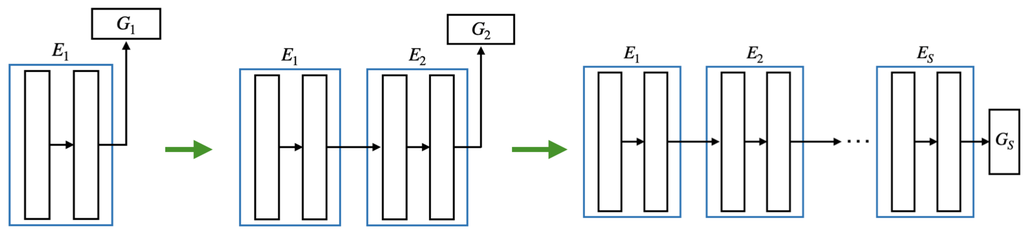

(a) Feed-forward networks




(b) U-nets (symmetric growing).

- **Training of Progressive Models**

To train the progressive model, we divide the original model $\mathcal{M}$ into disjoint features extractors $E_i$ and the classifier $G_S$ and extends the model every $T_s$ epochs. The temporal classifier $G_i$ will be discard when the model grows. We found that denoting roughly half of the total number of training iterations $T$ to progressive training, and setting $T_s = \frac{T}{2S}$ for $s < S$, $T_S = \frac{T(S+1)}{2S}$, such that $T = \sum_{s=1}^{S} T_s$, works well across all considered training tasks.




- **Asymptotic Rate**

We show the convergence rate and highlight that it is at most twice slower than the standard training due to the choice of the length of progressive training but with much cheaper per-iteration costs (empirically observed faster than the end-to-end training).

**Theorem** Let Assumptions 3.1 ($L$-smoothness) and 3.2 (bounded noise) 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:

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

- 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 $\frac{1}{T}\sum_{t=0}^{T-1} \|\nabla f(\mathbf{x}_t)\|^2 < \epsilon$ after at most the following iterations $T$:

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

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

## Experimental Results

ProgFed is more efficient than standard training with the same training epochs (also at any time during training) and compatible with existing compression and optimization.

- **Computation Efficiency**

Table 1:Results on CIFAR-100 in centralized settings.

| | 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% |

- **Communication Efficiency**

Table 2:Results in federated setups: Accuracy (%) and Dice scores (%), followed by cost reduction (CR) as compared to the 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 % |

- **Compatibility**

Table 3:Federated ResNet-18 on CIFAR-100 with (1) LQ-X denotes linear quantization followed by used bits and (2) SP-X denotes sparsification followed by ratios.

| | Float | LQ-8 | LQ-4 | LQ-2 | SP-25 | SP-10 | LQ-8 +SP-25 | LQ-8 +SP-10 |
|---|---|---|---|---|---|---|---|---|
| | Accuracy (%) | | | | | | | |
| Baseline | 52.08 | 49.40 | 49.55 | 47.26 | 51.23 | 51.79 | 49.67 | 50.25 |
| Ours | **53.23** | **53.07** | **52.32** | **52.87** | **52.00** | **51.86** | **52.19** | **52.24** |
| | Compression Cost (%) | | | | | | | |
| 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** |

Table 4:Results of ProgFed with FedAvg, FedProx, and FedAdam on CIFAR-100 in the federated setting.

| 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** |