On-Device Training with Local Sparsity for Federated Learning

Xinchi Qiu[1], Javier Fernandez-Marques[2], Pedro PB Gusmao[1], Yan Gao[1], Titouan Parcollet[3] and Nicholas D. Lane [1]
[1]University of Cambridge, [2]University of Oxford, [3]Université Avignon,

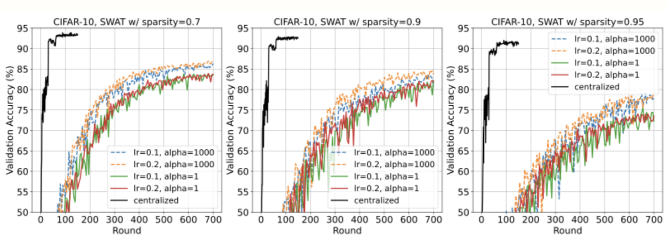
UNIVERSITY OF CAMBRIDGE | UNIVERSITY OF OXFORD | AVIGNON UNIVERSITÉ | EPSRC Engineering and Physical Sciences Research Council

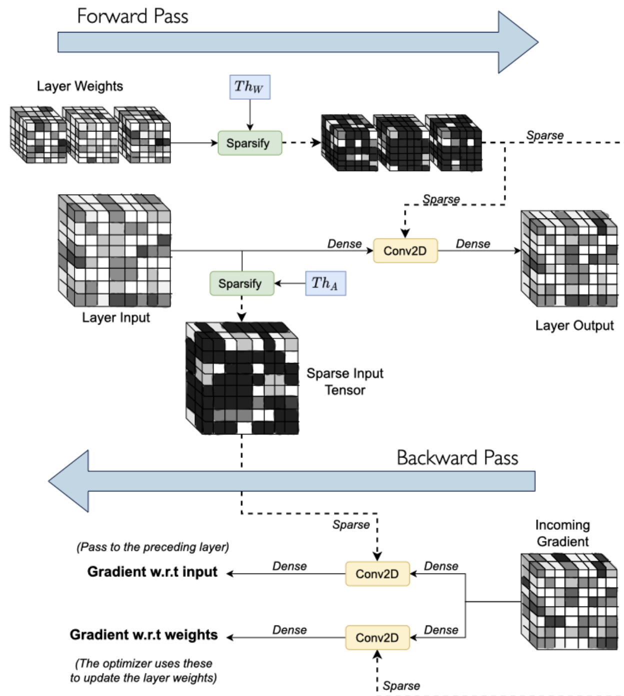
TL;DR

- We explore training with highly sparse tensors in FL clients to accelerate compute. We achieve this by replacing all dense convolutions with sparse-dense convolutions. These can be accelerated for a sufficiently high sparsity ratio.
- While sparse training has attracted some attention, it has not be investigated in the context of Federated Learning.
- While the resulting models are not sparse, we identify that the locations of high magnitude weights remains constant. We introduce a masking mechanism to exploit this observation and save on up-link communication.

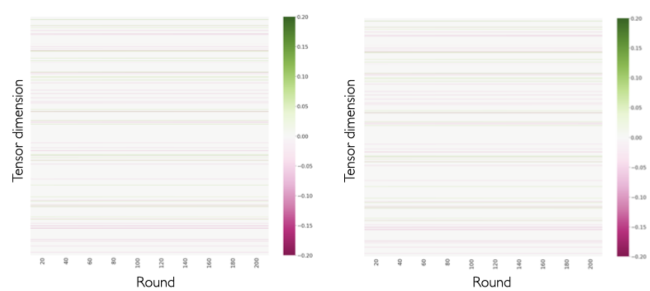
Federated Learning: Background and Challenges

- FL is a form of distributed ML where nodes are commodity devices such as smartphones, wearables or other IoT devices.
- The sophistication of the models that FL clients can train is constrained by the compute and memory limitations of the clients and the associated communications costs.
- These challenges has been partially eased by mechanisms relying on pruning, quantisation and distillation. These three techniques have dominated the recent "efficient FL" literature.
- Our method **replaces all dense convolutions with sparse-dense convolutions** in both forward and backward passes.
- We adapt SWAT (Raihan & Aamodt, 2020) to the FL setting and note that naively introducing sparsity result in **severe degradation compared to centralised training.**





- Only top-k weights are used during inference
- Non-zero weights at **constant locations** throughout the training process
- Only communicate **Top-(1-sp+mask ratio)** weights for aggregation



Algorithm 1 *ZeroFL*: Let us consider a cluster of N total client with n local data set and each with a learning rate η_t at round t with T the total number of communication rounds. The client has the data set n_k. The number of local epoch is E and the number of clients participating in each round is denoted as K. w_t represent all the weights aggregated at round t and d_t the difference of weights.

Central server does:
  for $t = 0, ..., T-1$ **do**
    Server randomly selects K devices.
    for all k in K **do**
      Perform TrainLocally(k, w_t)
    Aggregation:
    If Top-K-Weight then $w_{t+1} \leftarrow \sum_{k=0}^{K} \frac{n_k}{n} w_{t+1}^k$
    If Diff on Top-K-Weight then $w_{t+1} \leftarrow w_t + \sum_{k=0}^{K} \frac{n_k}{n} d_{t+1}^k$
    If Top-K Diff then $w_{t+1} \leftarrow w_t + \sum_{k=0}^{K} \frac{n_k}{n} d_{t+1}^k$
TrainLocally(k, w_t):
  for $e = 1, ..., E$ **do**
    Do local model training via SWAT with sparsity level sp.
    $w_e \leftarrow w_{e-1} - \eta_t \nabla F(w_{e-1})$
  Determine which weights to send for aggregation:
    If Top-K-Weight then return top $1 - sp + r_{mask}$ weights.
    $d_{t+1}^k \leftarrow w_E - w_t$
    If Diff on Top-K-Weight then return d_{t+1}^k of top $1 - sp + r_{mask}$ weights.
    If Top-K-Weights Diff then return top $1 - sp + r_{mask}$ of d_{t+1}^k.

We consider three masking mechanisms

Results

- Study the impact of ZeroFL sparse training and masking in both IID and non-IID settings for image classification (CIFAR-10, FEMNIST) and keyword spotting audio classification (Speech Commands)
- Masking balances communication and training/aggregation quality.
- We observe that high-magnitude weights remain at fixed location throughout training. By masking small values, the performance of the global model increases and communication savings are obtained.

non-IID	Sparsity Level	SWAT Full Model	ZeroFL (m=0.2)	File Size (MB)	Comms Save
CIFAR-10	90%	80.62%	81.04%	27.3	1.6x
	95%	74.00%	75.54%	23.0	1.9x
Speech Commands	90%	82.81%	84.90%	27.3	1.6x
	95%	81.12%	82.02%	23.0	1.9x
FEMNIST	95%	83.34%	83.78%	4.4	5.2x