FedSynth: Gradient Compression via Synthetic Data in Federated Learning

Carnegie Mellon University

Meta

Shengyuan Hu (CMU)*, Jack Goetz (Meta), Kshitiz Malik (Meta), Hongyuan Zhan (Meta), Zhe Liu (Meta), Yue Liu (Meta)

Background

- Training large models in Federated Learning (FL) suffers from huge communication cost. Therefore, information compression is important in the context of large scale FL.
- Prior works studying compression in FL focus on sparsification / quantization of gradients / model updates.
- Large compression rate under sparsification based compression might lead to drastic utility tradeoff.
- This work: proposes to distill the dataset and communicate the synthetic data used to reconstruct the gradient updates.

Compression via Synthetic Data

- Communicating data is more efficient than communicating model.
- Instead of sending a large model update, we can send synthetic data to the server such that the server could use the synthetic data to reconstruct an approximate model update.

Notations

- $D_k^{tr} = (X_k, Y_k)$: training data for client k
- $D_k^{syn} = \{x_k^i, y_k^i\}_{i=1,\cdots,m}$: m batches of synthetic data for client k

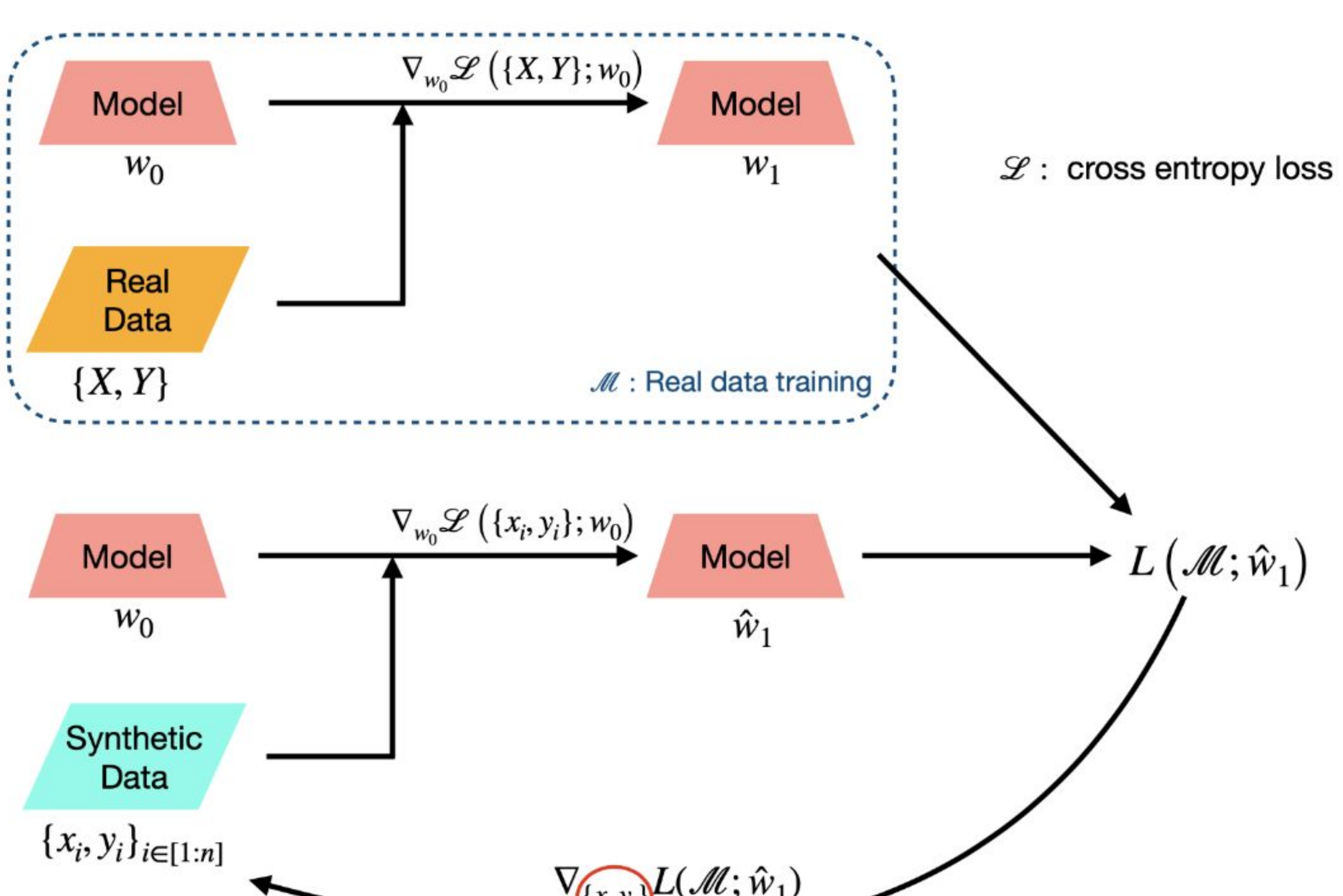
Local objective for standard FL

$$\min_w F_k(D_k^{tr}; w)$$

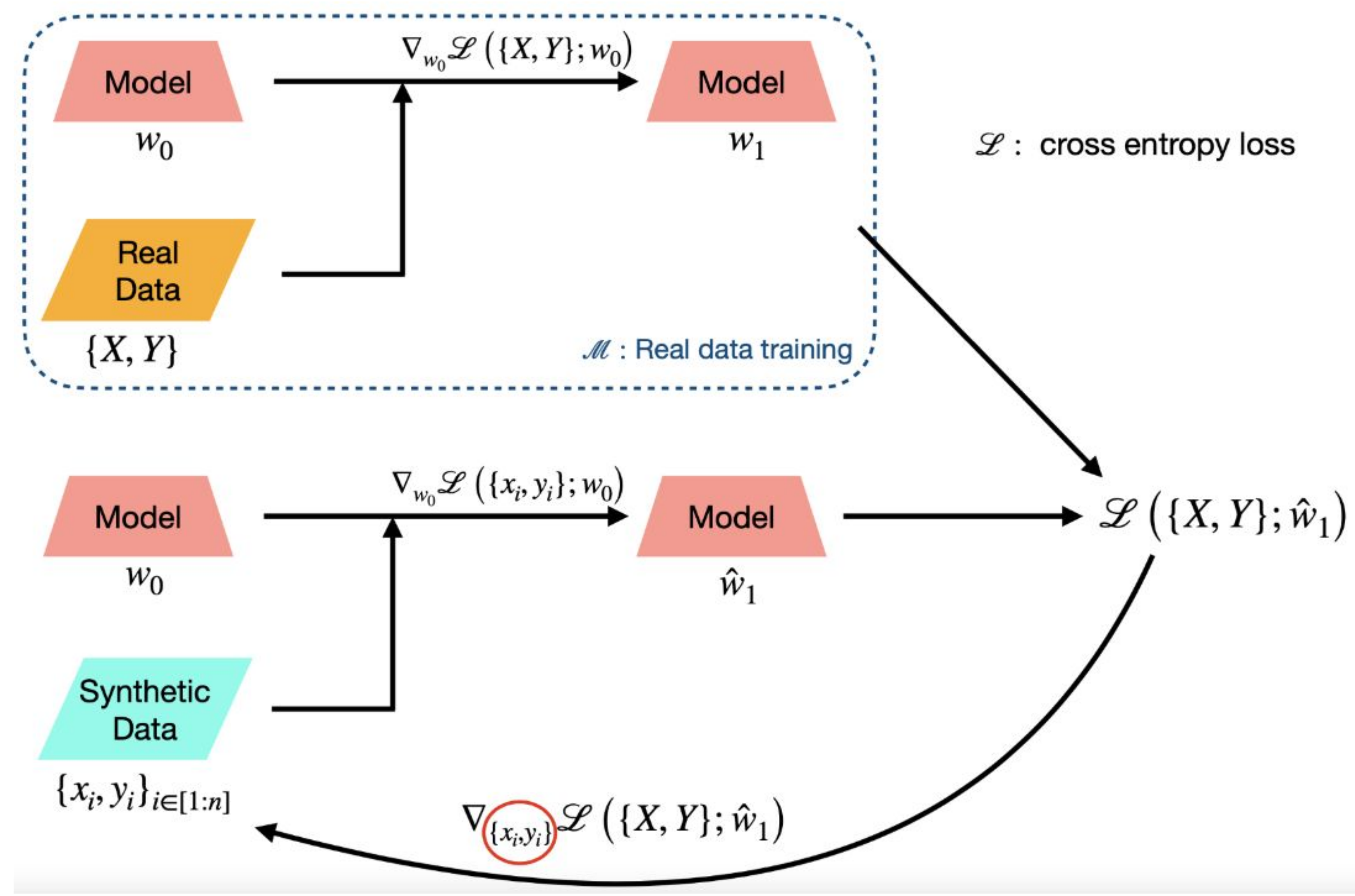
Local objective for FedSynth at any iteration

$$\min_{D_k^{syn}} F_k\left(D_k^{tr}; \arg\min_w F_k(D_k^{syn}; w)\right)$$

$$\min_{D_k^{syn}} F_k\left(D_k^{tr}; \text{ClientUpdate}_k(D_k^{syn}; w_k^t)\right)$$



*: Work done as an intern at Meta



Algorithm for FedSynth

Algorithm 1 FedSynth

Input: $T, E, \eta, \eta_w, w^0, \{D_k^{tr}\}_{k=1,\cdots,K}$
for $t = 0, \cdots, T-1$ **do**
  Server selects a subset of clients S_t and broadcasts w^t to S_t.
  for all $k \in S_t$ in parallel **do**
    Client k initializes $w_k^t = w^t$ and m batches of synthetic data $D_k^{syn} = \{x_i, y_i\}_{i=1,\cdots,m}$.
    for $j = 0, 1, \cdots, E$ **do**
      Client k obtains the model updated by D_k^{syn}
      $$w_k^{syn} = \texttt{ClientUpdate}(D_k^{syn}; w_k^t)$$
      Client k updates D_k^{syn} by
      $$D_k^{syn} \leftarrow D_k^{syn} - \eta \nabla_{D_k^{syn}} F_k(D_k^{tr}; w_k^{syn})$$
    end for
    Client k sends D_k^{syn} back to the server.
  end for
  Server recovers $\widehat{w}_k^{syn} = \texttt{ClientUpdate}(D_k^{syn}, w_k^t)$ for every k.
  Server aggregates the weight
  $$w^{t+1} = w^t + \frac{1}{|S_t|}\sum_{k\in S_t}(\widehat{w}_k^{syn} - w^t)$$
end for
return w^T

$\texttt{ClientUpdate}(\{x_i, y_i\}_{i=1,2,\cdots,m}; w)$
for $j = 1, \cdots, m$ **do**
  Client performs minibatch-SGD locally
  $$w \leftarrow w - \eta_w \nabla_w F_k((x_i, y_i); w)$$
end for

Experiment Results

FEMNIST	FedAvg	Random Masking	FedSynth(Ours)	FedSynth w/ Trainable y (Ours)
1x	69.29	69.29	69.29	69.29
5.8x	-	68.21	**68.63**	46.67
11.6x	-	**67.34**	63.27	39.98
MNIST	FedAvg	Random Masking	FedSynth(Ours)	FedSynth w/ Trainable y (Ours)
1x	97.74	97.74	97.74	97.74
7.8x	-	97.08	95.28	**97.25**
15.6x	-	**96.94**	93.68	96.62
Reddit	FedAvg	Random Masking	FedSynth(Ours)	FedSynth w/ Trainable y (Ours)
1x	14.19	14.19	14.19	14.19
1.3x	-	8.20	**8.86**	-
2.6x	-	4.87	**4.89**	-

- Our method is able to achieve comparable / better performance compared to random masking under all three datasets, especially under low compression rate.
- Our method with trainable label does not always give better utility given the same compression rate.

Future works

- Compare with stronger baselines on all three datasets.
- Experiment on larger models and more complicated tasks.