Evolutionary Federated Learning Using Particle Swarm Optimization

Ender Minyard[*]
Department of Applied Mathematics
Columbia University
cum2102@columbia.edu

Steven Kolawole*
ML Collective
steven@mlcollective.org

Nayan Saxena
ML Collective
nayan@mlcollective.org

Abstract

Communication efficiency is a first-order concern in centralized federated learning. Local computation reduction can improve the communication efficiency bottleneck of federated learning algorithms. We explore a novel federated learning algorithm that leverages an evolutionary algorithm, particle swarm optimization, to reduce local computation on federated learning clients. Our experimental results indicate improved accuracy and loss optimization in fewer time-steps as compared to other baselines.

1 Background

Federated Learning Federated learning is privacy-preserving machine learning that lets clients keep training data on personal devices while training models locally. In centralized federated learning, federated learning clients send model updates to a central server, as opposed to traditional centralized learning, where clients send personal training data to a server [1].

Evolutionary algorithms Evolutionary algorithms are biologically inspired optimization algorithms that approximate the minima of an objective. A generic evolutionary algorithm generates an initial population with random individuals, evaluates the fitness of individuals in the initial population, selects the fittest individual in a population according to an objective function, and then restarts this cycle, beginning at evaluation [2]. Evolutionary algorithms require fewer compute resources to optimize an objective than deterministic gradient-based optimization algorithms that guarantee convergence[3].

In this paper we use particle swarm optimization as our choice of evolutionary algorithm in conjunction with federated learning because it is computationally inexpensive with respect to both memory and speed [4]. Clients participating in a round of federated learning who train their model using particle swarm optimization instead of gradient descent will require fewer computational resources to train their local model.

2 Experimental Setup

Centralized Training We utilized knowledge transfer to reduce the local computation required on client devices [5]. We froze the first five layers of MobileNetV2 pre-trained with ImageNet weights

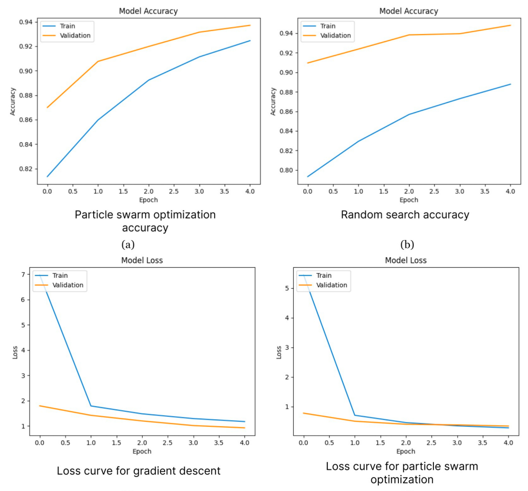
[*]Equal Contribution

Preprint. Under review.

(top not included). We transferred the weights to a new Keras model, after which we added several layers. The layers we added to the new architecture included a Flatten layer, a Dense layer, a ReLU activation function, a Dropout layer, and a Dense layer that had ten units and a softmax activation. We evaluate the model's accuracy at each location in the search space of hyperparameters with the Adam optimizer and a categorical cross-entropy loss function. Our dataset contained 60,000 MNIST training examples and 10,000 test examples. The experiments were run on the Intel Iris Plus Graphics 1536 MB GPU.

Federated Setting We simulated federated evolution in a client-server architecture using TensorFlow Federated[6] and Flower[7] as our training frameworks, and we trained using the EMNIST[8] and FEMNIST[9] datasets. We use stochastic weight averaging [10], to aggregate weights on the server side.

3 Results



Particle swarm optimization accuracy
(a)

Random search accuracy
(b)

Loss curve for gradient descent
(c)

Loss curve for particle swarm optimization
(d)

Figure 1: Comparison of model accuracy and loss over time between training (blue) and validation sets (orange) for particle swarm optimization, Fig. 1 (a) & (c), and random search, Fig. 1 (b) & (d).

Optimization	Time	Accuracy
Particle swarm	86.1671	0.9624
Gradient descent	52.6553	0.6814
Random search	44.4513	0.9501

Table 1: Comparison of optimization time and accuracy for different approaches.

As shown in Fig. 1, an early decline in the loss curve for particle swarm optimization is observed as compared to gradient descent. Furthermore, as observed in Table 1, particle swarm optimization performed better than gradient descent when it comes to accuracy.

4 Discussion

Our exploration of particle swarm optimization demonstrates that evolutionary approaches have the potential to reduce local computation in federated learning. This could allow us to improve the communication efficiency of federated learning by reducing the rounds of communication needed to train a shared model. A direct next step is to now replicate these experiments through client-server simulations where similar performance is hypothesised. Furthermore, with approaches such as accelerated particle swarm optimization [11], we believe our proposed method could be sped up even further. From a software standpoint, through our experiments, we also observed that TensorFlow

Federated is not designed to work with custom optimization algorithms. Moreover, (de)-serialization of weights is tied to how the federated weights are trained in Flower; hence leading to issues communicating the serialized weights between the clients and the server. Future work can focus on tackling these setbacks, to achieve a client-server architecture designed explicitly for evolution-based federated learning. Our work also opens up the opportunity to incorporate privacy preserving techniques within our framework in the future.

References

[1] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency, 2016. URL `https://arxiv.org/abs/1610.05492`.

[2] P. A. Vikhar. Evolutionary algorithms: A critical review and its future prospects. *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, pages 261–265, 2016.

[3] Bayat Hamed, Akbari Hassan, and Davoudpour Hamid. A metaheuristic and ptas approach for np-hard scheduling problem with controllable processing times. volume 3, pages 307–314. Academic Journals, 2010.

[4] Riccardo Poli, James Kennedy, and Tim M. Blackwell. Particle swarm optimization. *Swarm Intelligence*, 1:33–57, 1995.

[5] Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group knowledge transfer: Federated learning of large cnns at the edge, 2020. URL `https://arxiv.org/abs/2007.14513`.

[6] K Bonawitz, H Eichner, W Grieskamp, et al. Tensorflow federated: machine learning on decentralized data., 2020.

[7] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, Pedro PB de Gusmão, and Nicholas D Lane. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

[8] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[9] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[10] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

[11] Amir Hossein Gandomi, Gun Jin Yun, Xin-She Yang, and Siamak Talatahari. Chaos-enhanced accelerated particle swarm optimization. *Communications in Nonlinear Science and Numerical Simulation*, 18(2):327–340, 2013.