**Federated Learning - TensorFlow EMNIST**

**Krishnaveni Mandadi(kmandadi@memphi.edu)**  
Masters in Data Science, University of Memphis, TN, 38111, USA, 08-04-2022

**Introduction**:

Federated training is a shared global model which is trained across the locally trained models. Locally trained models are training the machine learning models on the devices such as mobiles, tabs, and personal computers on quality data and without uploading sensitive typing data to servers.

Federated machine learning has a few distinct advantages to centralized machine learning, including:[2]

* Protecting user privacy because the data is stored on the user's device.
* Lower latency since predictions can be made on the user's device using the updated model.
* Given the collaborative training process, smarter models.

**Problem Statement:**

The canonical federated learning problem involves learning a single, global statistical model from data stored on tens to potentially millions of remote devices. We aim to learn this model under the constraint that device-generated data is stored and processed locally, with only intermediate updates being communicated periodically with a central server. In particular, the goal is typically to minimize the following objective function [1].

min *F*(*w*) , where *F*(*w*):= ∑*PkFk*(*w*)

*w k=1*

Here, m is the total number of devices, *Pk* ≥ 0 and ∑k *Pk*= 1, and Fk is the local objective function for the kth device. The local objective function is often defined as the empirical risk over local data, i.e., *Fk* (*w*) = 1/n*k* ∑*nkfj* (*w*; *xj* , *yj* ), where n*k* is the number of samples available locally. The user-defined term pk specifies the relative impact of each device, with two natural settings being pk = 1/n or *Pk* = n*k*/n , where n = ∑kn*k* is the total number of samples. We will reference problem (1) throughout the article, but, as discussed below, we note that other objectives or modeling approaches may be appropriate depending on the application of interest.

**Model Implementation:**

TFF enables developers to simulate the included federated learning algorithms on their models and data, as well as to experiment with novel algorithms. Researchers will find [starting points and complete examples](https://www.tensorflow.org/federated/tff_for_research) for many kinds of research. The building blocks provided by TFF can also be used to implement non-learning computations, such as [federated analytics](https://ai.googleblog.com/2020/05/federated-analytics-collaborative-data.html). [5] TFF’s interfaces are organized in two main layers:

Federated Learning (FL) API

This layer offers a set of high-level interfaces that allow developers to apply the included implementations of federated training and evaluation to their existing TensorFlow models.

Federated Core (FC) API

At the core of the system is a set of lower-level interfaces for concisely expressing novel federated algorithms by combining TensorFlow with distributed communication operators within a strongly-typed functional programming environment. This layer also serves as the foundation upon which we've built Federated Learning.

**FL API**

The interfaces offered by this layer consist of the following three key parts:

**Models**. Classes and helper functions that allow you to wrap your existing models for use with TFF. Wrapping a model can be as simple as calling a single wrapping function (e.g., [tff.learning.from\_keras\_model](https://www.tensorflow.org/federated/api_docs/python/tff/learning/from_keras_model)), or defining a subclass of the [tff.learning.Model](https://www.tensorflow.org/federated/api_docs/python/tff/learning/Model) interface for full customizability.

**Federated Computation Builders**. Helper functions that construct federated computations for training or evaluation, using your existing models.

**Datasets**. Canned collections of data that you can download and access in Python for use in simulating federated learning scenarios. Although federated learning is designed for use with decentralized data that cannot be simply downloaded at a centralized location, at the research and development stages it is often convenient to conduct initial experiments using data that can be downloaded and manipulated locally, especially for developers who might be new to the approach.

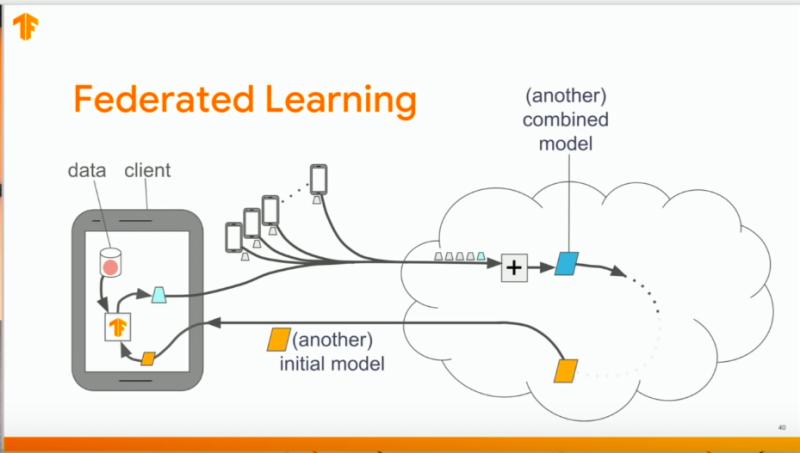
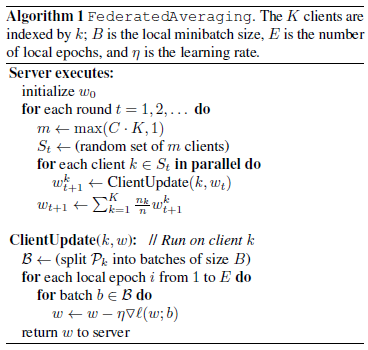


Fig1. Federated Learning architecture[2]

A typical round of learning consists of the following sequence: [7]

* A random subset of members of the Federation (known as clients) is selected to receive the global model synchronously from the server.
* Each selected client computes an updated model using its local data.
* The model updates are sent from the selected clients to the server.
* The server aggregates these model weights (typically by averaging) to construct an improved global model.



pseduo code of federated learning[7]

**Experiment and Results:**

**Dataset,** Tensorflow authors seeded the TFF repository with a few datasets to allow for experimentation, including a federated version of MNIST that includes a copy of the original NIST dataset that has been re-processed with Leaf so that the data is keyed by the person who wrote the digits in the first place. Each writer has their own distinct style, hence this dataset displays the non-i.i.d. behavior anticipated of federated datasets.

Since the data is already a [tf.data.Dataset](https://www.tensorflow.org/api_docs/python/tf/data/Dataset), preprocessing can be accomplished using Dataset transformations. Here, we flatten the 28x28 images into 784-element arrays, shuffle the individual examples, organize them into batches, and rename the features from pixels and label to x and y for use with Keras. We also throw in a repeat over the data set to run several epochs.[6]

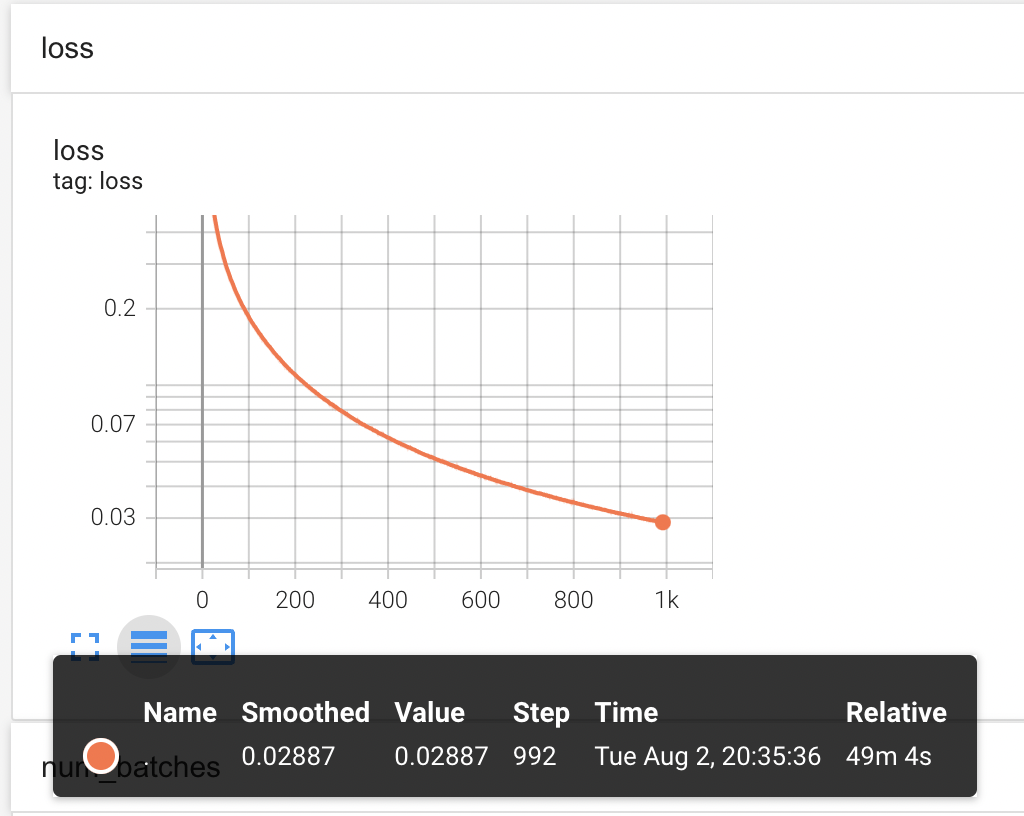
A simple Python list can be used to feed federated data to TFF in a simulation. Each element of the list contains the data for a individual user, either as a list or as a tf.data.Dataset. Any model must be wrapped in an instance of the tff.learning in order to be used with TFF. Similar to Keras, the model interface includes methods for stamping the model's forward pass, metadata properties, etc., but also adds new features like controls over how federated metrics are computed.

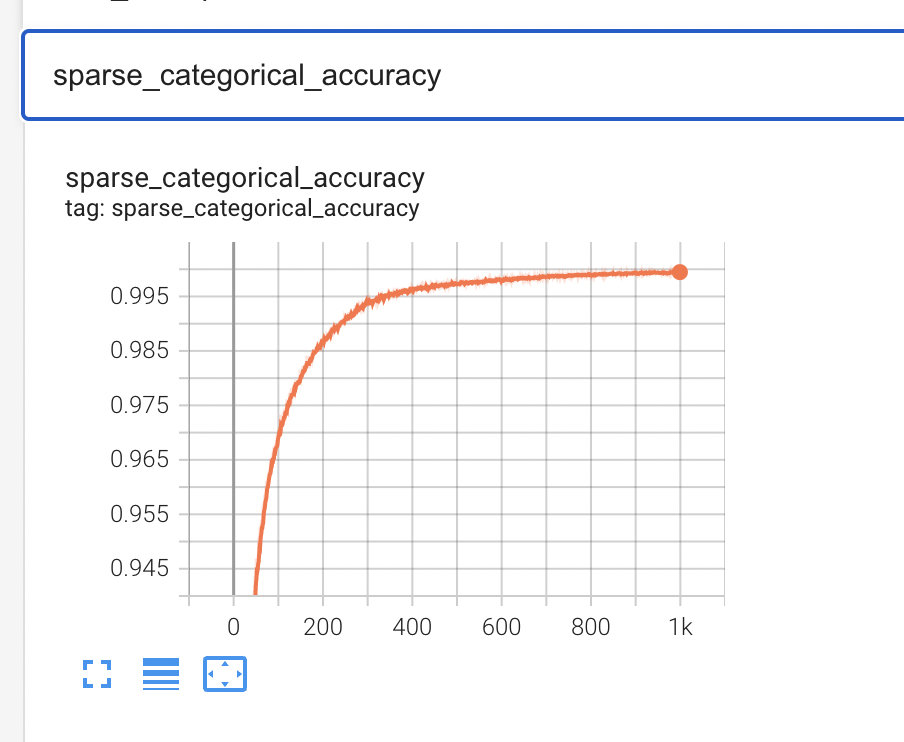
Now that we have a model wrapped as [tff.learning.Model](https://www.tensorflow.org/federated/api_docs/python/tff/learning/Model) for use with TFF, we can let TFF construct a Federated Averaging algorithm by invoking the helper function [tff.learning.algorithms.build\_weighted\_fed\_avg](https://www.tensorflow.org/federated/api_docs/python/tff/learning/algorithms/build_weighted_fed_avg). Let's run a single round of training and visualize the results. We can use the federated data that already generated above for a sample of users and repeat the process for multiple rounds.

Now construct a Federated Averaging iterative process, which we will use to improve the model.

The initial state of the model produced by fed\_avg.initialize() is based on the random initializers for the Keras model, not the weights that were loaded, since clone\_model() does not clone the weights. To start training from a pre-trained model, we set the model weights in the server state directly from the loaded model.

All of the experiments so far presented only federated training metrics - the average metrics over all batches of data trained across all clients in the round.





Further, the results contain the loss and accuracy parameters for ten iterations repeated for SGD optimizer for each dataset. The peak accuracy that we achieved for E-MNIST datasets is 97% on training data and 90% on test data by using federated averaging algorithm. Further, the minimum loss value that we obtained for E-MNIST datasets 0.1895227.

Training loss is decreasing after each round of federated training, indicating the model is converging. There are some important caveats with these training metrics, and I have set the training to go for 250 rounds. The training loss at the end of the training is 0. 1895227 down from 2.740445 recorded at the start of the training. we do not have each worker generating its gradient and writing to the parameter server. there are 2 optimizers: a \_clientoptimizer and a \_serveroptimizer. The \_clientoptimizer is only used to compute local model updates on each client. The \_serveroptimizer applies the averaged update to the global model at the server. The training time for each worker node here was noted to be 16.75 seconds.

I am in the process of exploring more about federated learning and the information provided on this document is from extensive survey from different articles and recent experiments on image classification and text generation by tensorflow authors.

**References:**

[1]Federated Learning: Challenges, Methods, and Future Directions[Tian Li, Anit Kumar Sahu, Ameet Talwalkar, Virginia Smith]

[2][Basil Han](https://medium.com/@hanbasil?source=post_page-----8a1a62b0600d--------------------------------) An Overview of Federated Learning, A look at its history, potential, progress, and challenges <https://medium.datadriveninvestor.com/an-overview-of-federated-learning-8a1a62b0600d>

[3]Federated Learning: An Introduction, Improving machine learning models and making them more secure by training on decentralized data. By [Derrick Mwiti](https://www.kdnuggets.com/author/derrick-mwiti), Data Scientist on April 15, 2020 in [Federated Learning](https://www.kdnuggets.com/tag/federated-learning), [Learning](https://www.kdnuggets.com/tag/learning), [Machine Learning](Machine%20Learning), [Privacy](https://www.kdnuggets.com/tag/privacy), [Security](https://www.kdnuggets.com/tag/security) <https://www.kdnuggets.com/2020/04/federated-learning-introduction.html>

[4]Tensorflow federated: Machine learning on decentralized data. URL://www.tensorflow.org/federated/federated\_learning

[5]Tensorflow federated: URL <https://www.tensorflow.org/federated>

# [6] Federated Learning for Image Classification <https://www.tensorflow.org/federated/tutorials/federated_learning_for_image_classification>

[7]Communication-Efficient Learning of Deep Networks from Decentralized Data [H. Brendan McMahan](https://arxiv.org/search/cs?searchtype=author&query=McMahan%2C+H+B), [Eider Moore](https://arxiv.org/search/cs?searchtype=author&query=Moore%2C+E), [Daniel Ramage](https://arxiv.org/search/cs?searchtype=author&query=Ramage%2C+D), [Seth Hampson](https://arxiv.org/search/cs?searchtype=author&query=Hampson%2C+S), [Blaise Agüera y Arcas](https://arxiv.org/search/cs?searchtype=author&query=Arcas%2C+B+A+y)