FEDAVG AND 1D CONVOLUTIONAL NEURAL NETWORKS FOR PRIVATE AND EFFICIENT HUMAN ACTIVITY RECOGNITION

Olli Glorioso, olli.glorioso8@gmail.com

ABSTRACT

This report presents a solution to train a model at smartphone devices to recognize human activity based on smartphone acceleration sensor, and then to use these device models to privately create a global ML model to do same predictions.

The chosen federated algorithm for the problem is FedAvg, which as per its name averages the weights of the local models of its network. The network structure in this application is edgeless, since the global models picks the models to average over all of the nodes in the FL graph. The variation measure between the local models is averaging.

Keywords: Federated learning, networks, personalized machine learning, trustworthy artificial intelligence

1. INTRODUCTION

Measuring your bodily data and providing you with useful data visualizations is ubiquitos in this day. Smart watches, phones, even laptops are equipped with inertial measurement units. With the gyroscope and accelometer data from these sensors, the devices can determines what is happening to the user or what they are doing. For example, Apple Watch got recently a feature called "Fall Detection", where it can detect if the user has fell hard, and then further contact the emergency services [3].

However, as health data is very private and sensitive, most of such users do not want to share their data to external parties. On the other hand, having more data would increase the reliability of the reports and would also take the exact environment around the user better into account. Federated Learning offers a solution, where the users can train their models collaboratively across multiple devices without sharing raw data.

The state-of-the-art methods used in this report include Federated Learning Gradient Descent with parameter sharing to the neighbours as the main federated learning algorithm, 1D Convolutional Neural Networks as the local models, non-linear time-series accelerometer data, alpha-balanced Focal Loss categorical cross-entropy loss function for the local model optimization, and a Personal Federated Learning approach with a custom variation measure term.

The outline of the report is the following: Section 2 formulates the problem in terms of the devices with their local models in the FL network, Section 3 introduces the main methods to solve the formulated problem with a chosen parametric model variation measure and an FL algorithm. Section 4 explains the numerical experiments, including everything from introducing the chosen data source, model validation and the results themselves. Finally, Section 5 concludes the report and suggests future improvements or research topics.

2. PROBLEM FORMULATION

The FL application for the problem can be viewed as a network of nodes, their local models, loss functions, and the edges connecting the nodes. The network can be modeled formally as a graph tuple $G = V, E, A$, where V is the set of nodes, $E \subseteq V \times V$ is the set of edges, and $A : E \to \mathbb{N}^+$ is a function assigning weights to each edge.

The nodes in this application are the different users, specifically their phones that have accelerometers. They store a dataset X_i, which contain all the data they use to train their local models. The dataset contains historical data of the phone accelometers. Moreover, they store local CNN models defined by θ_i.

The ML models used at each node are 1D Convolutional Neural Networks (CNNs) specialized for a multi-class classification task. They will have three 1D Convolutional layers, one flattening layer, and an output dense layer with a softmax activation function. CNNs were chosen for this task as there is certainly no linear relation between the features of the chosen dataset. CNNs are generally also good for solving timeseries classification [4]. The convolution layers are one-dimensional, since the input data is also a one-dimensional time-series sequence. With 1D convolutions, the convolution kernel slides along the single spatial dimension of the time-series data [5].

The choice of loss function is categorical cross-entropy. There is two reasons for this choice. First one being that the chosen Python ML framework supports it out-of-the-box [6]. Second reason is that cross-entropy is a common method for measuring the loss in classification tasks, and categorical cross-entropy loss allows measuring the loss with multiple categories. In the sample dataset there are 18 categories. In the cho-

sen ML framework, this is an alpha-balanced focal loss [7]:

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{if } y = 0 \end{cases}$$

$$\text{FL}(p_t) = -\alpha \left(1 - p_t\right)^{\gamma} \log(p_t)$$

where γ is the focusing parameter that adjusts the rate at which easy examples are down-weighted, γ is the balancing factor that compensates for class imbalance.

In an optimal case, the edges and their weights would be chosen to connect individuals with similar physical conditions and capabilities. This way, the local models are optimized with the weights of similar models.

However, because the data set of this report includes only accelerometer data, it is infeasible to divide the devices into groups with similar behavior. Moreover, as this project will use FedAvg-algorithm, the decisions on edges between the nodes is not necessary as they will not affect the training process by any means.

3. METHODS

After experimenting with FedSG and failing to implement it with both Tensorflow- and PyTorch-frameworks, the choice of FL algorithm in this project is FedAvg for parametric local models. That is, a training loop is processed while a desired stopping criterion is not met. In this case, the stopping criterion would that the accuracy of the global model on its dedicated test set is over certain threshold m (see Chapter 4), which is formally the following:

$$\text{acc} := \frac{1}{n} \sum_{j=1}^{n} \mathbf{1}\left[\hat{y}_j = y_j\right] < m$$

where $\hat{y}_j$ is the predicted classification and y_j is the true classification. As the chosen FL algorithm is FedAvg, there is no variation measure between the nodes because all of the nodes are trained separately without taking the neighboring nodes into account. Just the global model gets its weights by averaging the weights of the local models.

The training loop includes the following steps, with k being the current iteration and $\mathbf{w}^k$ being the current global model parameters:

1. Randomly select a subset C^k of clients
2. Send the global CNN weights to all clients $c \in C^k$
3. Train the local models with their local datasets and the global CNN weights, obtaining $\mathbf{w}(c)$ for each client c
4. Update the the global to be the average of the parameters of the local clients' parameters:

$$\mathbf{w}^k := \frac{1}{|C(k)|} \sum_{i \in C(k)} \mathbf{w}(c).$$

where $w(c)$ are the trained weights of the local model at client c.

The previous algorithm mentions training the local weights. This is done in r local epochs (see Chapter 4) with the formula:

$$\mathbf{w}^{(0)} = \mathbf{w}_{\text{global}},$$
$$\mathbf{w}^{(r)} := \mathbf{w}^{(r-1)} - \eta \mathcal{L}_k(\mathbf{w}^{(r-1)}).$$

where r is the current local epoch and η is a globally chosen learning rate (see Chapter 4) and $\mathcal{L}_k\left(\mathbf{w}^{(r-1)}\right)$ is the focal loss function determined in Chapter 2.

The message passing in FedAvg is simple, but the specific case with CNNs needs some clarifications. The messages to pass are of format $\mathbf{w} = [[x_{1,0}, \ldots, x_{1,y}]^T, \ldots [x_{n,0}, \ldots, x_{4,z}]^T]$, where the sublists of the whole list are the weights of the different CNN layers and where the number of trainable CNN layers is n.

To conclude, the numerical experiments will be parametrized with the following variables: m for stopping criterion threshold, k_{max} for the maximum number of iterations, r for the number of local training epochs, and η for the global learning rate.

4. NUMERICAL EXPERIMENTS

The main data source of this report is the WISDM Smartphone and Smartwatch Activity and Biometrics Dataset [8]. Only the smartphone accelerator data is used in the experiment, as that itself already contains 4804403 datapoints for all the nodes in total. The dataset consists of the smartphone accelerator data for 51 test subjects as they perform 18 activities for 3 minutes apiece. Each datapoint has a subject-id between 1600-1650 identifying the test subject, and three features x, y, z representing the sensor reading for the respective dimensions. The label for each datapoint is an activity-code, which is a character from the set of characters from A to R, where each character defines the activity. Some examples of the activities include walking, jogging, brushing teeth, and eating sandwich. Refer to [9] for a full list of activities.

Furthermore, since the dataset contains timeseries data, it has to be batched into smaller datapoints consisting of a chosen number of timestamps with their respective features. In the experiments, a window size for the datapoints is 25, whereas the overlap size between two datapoints is 10. That is, each datapoint contains 25 datapoints and a single label for them, and neighbouring dataset share 10 datapoints each time.

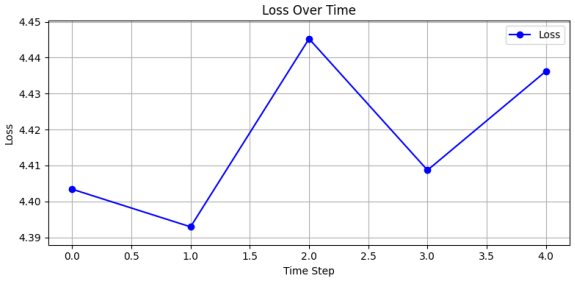

Fig. 1: Round 1 - Global loss on test dataset.

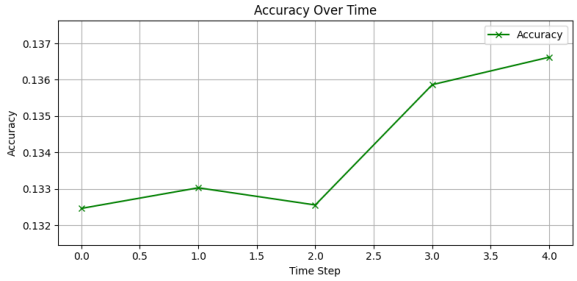

Fig. 2: Round 1 - Global accuracy on test dataset.

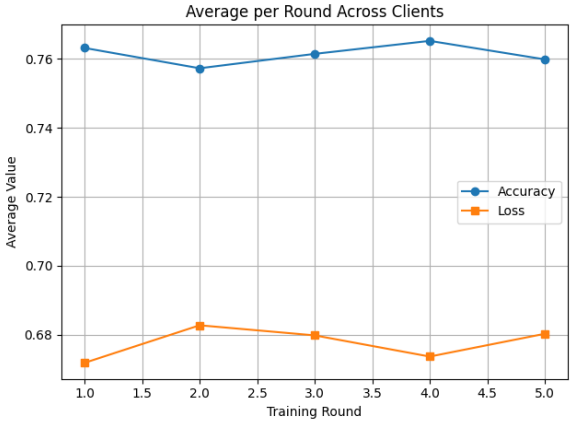

Fig. 3: Round 0 - Mean accuracies and losses across clients.

The batches are chosen so that each original datapoint in a same batch has the same label.

The local datasets are divided into three distinct sets: training sets, validation sets, and test sets. By manually choosing different dataset sizes, the chosen split is 70-15-15 for train-, validation-, and test-datasets respectively. Furthermore, 10% of each training set on each local model was also put aside for global test set for the final global model, so actually the training test set was 63% of the size of each local dataset. This is particularly viable for the chosen dataset, as it is quite big - it has around 94000 batched data points per node. During experimentation, I attempted to reduce the size of the local datasets to shorten the training time. However, after testing with sample sizes of 0.1 and 0.7, I observed a significant drop in accuracy, with a decrease of around 0.15. This trade-off in accuracy was too substantial to justify the reduced runtime.

In this experiment, the model selection is pretty simple - the global model is chosen to be the final model to benchmark against. It could be possible that one of the local models would be performing better than the global model in the early stages of the training, but overrall the global will be probably performing the best on the global test dataset. Most importantly, the local models will overfit to their own data. This makes them predict with a very high accuracy on their own local datasets, but this would not be the case with the global test dataset.

The first round of numerical experiments were conducted with the following parameter values: $k_{max} = 5$ to end the training early since the training process took a long time with the big dataset of this project, $r = 3$ to locally train the models somewhat extensively, and $\eta = 0.01$ as a learning rate at local models. Running even just 5 rounds of training took 143 minutes. Moreover, the first experiments did not choose a random subset of clients for each iteration - they just included all the clients. See second round of numerical experiments later for the results with random subsets. The results were not informative but could imply that averaging the global weights over time improves the accuracy of the model. As only five rounds of training was taken, the results can be quite randomly distributed and no big conclusions can be made.

First, the process of the global model against the test dataset. The focal loss was stayed mostly the same without any growth to any direction, staying between values 4.39 and 4.45 (Fig. 1). However, the accuracy seemed to improve in very small steps, starting from around 0.132 and increasing up to 0.137 (Fig. 2).

The accuracy at the local models is expectedly way higher than with the global model, since they are fitting to their own datasets. The means of all of the clients' losses and accuracies on the validation datasets over the 5 rounds are quite randomly distributed, and the accuracy stays at around 0.75 and the loss stays at 0.68 (Fig. 3).

The second round of numerical experiments were done with different parametrization. The experiments were conducted with the same learning rate, $k_{max} = 17$, $r = 1$ and $\eta = 0.01$. Hence, the maximum number of rounds was four times higher but there was only one local epoch at each client. Moreover, 10 clients were randomly picked at each iteration for global weight averaging, instead of using all of the clients.

With the second round, along with different sort of parametrization, a more linear pattern can be seen in the improvement of the global model accuracy. The global model improves its accuracy near linearly, at least in the beginning of the training process (Fig. 4). Furthermore, also the loss improved at least from the beginning of the process (Fig. 5). Overall, the av-

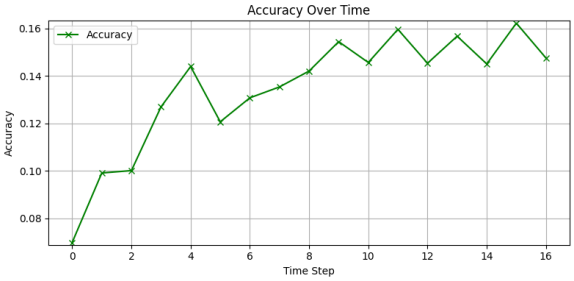

Fig. 4: Round 2 - Global accuracy on test dataset.

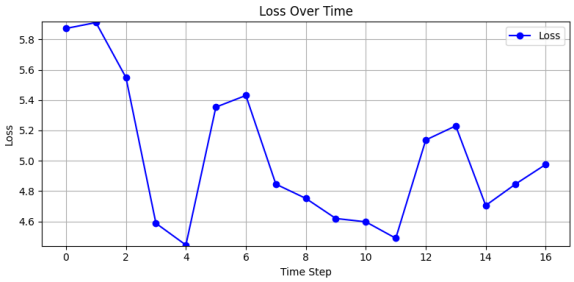

Fig. 5: Round 2 - Global accuracy on test dataset.

erage of validation losses and accuracies and models kept their average similar to the ones from the round 1.

5. CONCLUSION

Firstly, let's analyze the results of the first training round. Since the dataset was so large and it was not an option to reduce the local dataset sizes, the training time took inconveniently long and the numerical experiments only consisted of 5 rounds with very slow accuracy progression. Because of this, the obtained results do not yet solve the problem satisfactorily. There are two options for further experimenting.

Firstly, the training time should be decreased by either choosing a different dataset with less datapoints or by sampling the datapoints' before the training progress. This way, more rounds of the iteration could be done and thus the global weights could become more accurate.

The second option is to change the federated learning algorithm from FedAvg to FedGD, which would optimize the local models instead of creating a global model that tries to fit to all of the datapoints. This was the original idea, but could not be implemented because of the inability to create the Tensorflow infrastructure for it.

The second round of training was more successful as the accuracy grow in somewhat linear manner compared to the iteration number. So probably implementing the random subset picking and averaging those weights is a better option for FedAvg compared to using all the models in the network.

6. REFERENCES

[1] A. Jung, *Federated Learning: From Theory to Practice*, Aalto, 2025. Available: https://github.com/alexjungaalto/FederatedLearning/blob/main/material/FLBook.pdf.

[2] A. Jung, *Machine Learning: The Basics*, Springer, 2022.

[3] Apple Inc., *Use Fall Detection with Apple Watch*, 2025. Available: https://support.apple.com/en-us/108896.

[4] Wibawa, A.P., Utama, A.B.P., Elmunsyah, H. et al. *Time-series analysis with smoothed Convolutional Neural Network*, J Big Data 9, 2021. Available: https://doi.org/10.1186/s40537-022-00599-y

[5] Thakur, A., *Intuitive understanding of 1D, 2D, and 3D convolutions in convolutional neural networks.*, 2024. Available: https://wandb.ai/ayush-thakur/dl-question-bank/reports

[6] Tensorflow, *tf.keras.losses.CategoricalCrossentropy*, 2025. Available: https://www.tensorflow.org/api_docs/python/tf/keras/losses/CategoricalCrossentropy

[7] Keras Team, *Keras 3: Depp Learning for Humans*, 2025. Available: https://github.com/keras-team/keras/tree/v3.3.3

[8] Weiss, G., *WISDM Smartphone and Smartwatch Activity and Biometrics Dataset*, 2019. Available: https://doi.org/10.24432/C5HK59

[9] Weiss, G., *WISDM Dataset Description*, 2019. Available: https://archive.ics.uci.edu/ml/machine-learning-databases/00507/WISDM-dataset-description.pdf