

# Federated Learning for Collaborative Prognosis

Maharshi Dhada<sup>1\*</sup>, Adria Salvador Palau<sup>1</sup>, Ajith Kumar Parlikad<sup>1</sup>

<sup>1</sup>Department of Engineering, Institute for Manufacturing, University of Cambridge, Cambridge, CB3 0FS, UK

\*Corresponding author: mhd37@cam.ac.uk

**Abstract:** Modern industrial assets generate prodigious condition monitoring data. Various prognosis techniques can use this data to predict the asset's remaining useful life. But the data in most asset fleets is distributed across multiple assets, bound by the privacy policies of the operators, and often legally protected. Such peculiar characteristics make data-driven prognosis an interesting problem. In this paper, we propose Federated Learning as a solution to the above mentioned challenges. Federated Learning enables the manufacturer to utilise condition monitoring data without moving it away from the corresponding assets. Concretely, we demonstrate Federated Averaging algorithm to train feed-forward, and recurrent neural networks for predicting failures in a simulated turbofan fleet. We also analyse the dependence of prediction quality on the various learning parameters.

**Keywords:** *Smart Manufacturing, Federated Learning, Fleet Prognostics, Collaborative Prognostics*

## 1. Introduction and Research Gap

The recent surge in sensor technologies has enabled the generation of real-time condition monitoring data from industrial assets [1]. Various Machine Learning (ML) algorithms can analyse this data and predict the Remaining Useful Life (RUL) of the assets, this is called data-driven *prognosis*. Prognosis is important because an accurate RUL prediction helps forming an efficient maintenance plan [2]. As a result, the overall life cycle costs of the assets are reduced.

Prognosis relies on predictive models generated by the ML algorithms trained using an asset's time-series data ranging from the healthy state of the asset till its failure. For a single asset, this training data is gathered from the asset's history of failures. In addition to its own failures, an asset in the fleet can also learn from the failures of other assets. This is especially helpful in scenarios where the individual asset data is not sufficient to generate a confident prognosis model. But since the assets are not identical, the data has to be carefully selected so that the participating assets are not too different [3].

*Collaborative prognosis* is the state-of-the-art technique to enable inter asset learning [3]. Assets are represented by corresponding computational agents running predictive models for their RUL prediction. The data originating from each asset is compared with that of others to identify assets operating in similar conditions. The amount of data shared between the assets is subsequently decided based on their similarity. It has been shown that such collaborative prognosis in theory is more cost effective compared to self learning (prognosis using the machine's own data), and also learning from other assets' data directly [3].

However, the process of training ML algorithms online requires modifications before collaborative prognosis can

actually be implemented for real world industries. This is detailed in the following paragraph.

Collaborative prognosis currently requires the assets comprising the fleet to share data with one another. Such data sharing is not feasible for the assets in a fleet that are owned by different operators who would not wish to share their data among each other. E.g. Gas turbines owned by different companies [4]. Data transfer also rapidly increases the communications costs for the manufacturer [5]. Prognosis algorithms therefore need to be designed so that the data distributed across a fleet can be efficiently utilised [6].

This paper explores the potential of Federated Learning (FL) [7] as a possible solution to address these challenges of implementing collaborative prognosis. FL involves training the prediction models without moving the data away from the corresponding assets where it is generated. In this paper we demonstrate the use of *Federated Averaging* (FedAvg) algorithm [8] to train various neural network models for predicting failures in a fleet of turbofans. The dependence of model convergence on different learning hyperparameters is also analysed.

**Section 2** discusses FL in greater length, and presents the supporting mathematics for FedAvg algorithm. We used the recently released TensorFlow Federated (TFF) library [9] for our simulations, which is described in **Section 3**. The results arising from empirical explorations are shown in **Section 4**, and the inferred conclusions comprise **Section 5**. Possible directions to further this work are discussed in **Section 6**.

## 2. Federated Learning

### 2.1 Origin and definition

Federated Learning (FL) originated to address the problem of training predictive models in mobile devices

[8]. The devices are connected to one another via a server, thus forming a client-server network setup. Similar to RUL prediction in a fleet of assets, the performance of ML algorithms used in mobile devices can be improved by training them using data from other similar users. But the privacy of the users must be respected, and the data therefore cannot be shared within the users. FL enables training the predictive models under these constraints by pushing the training process to the clients, rather than the server [8]. A detailed description of FL can be found in [7].

In our experiments we demonstrate the use of a naïve FL algorithm called **FederatedAveraging (FedAvg)** to train a RUL prediction model for assets operating in similar conditions. This algorithm is capable of replacing the data sharing step of collaborative prognosis.

In FedAvg, the server stores a global predictive model which is passed on to a randomly selected subset of clients (called federation) after fixed time-steps. Each client in the federation trains the model using its local data, and sends the parameter updates back to the server. The server accumulates these updates and generates a new global model. The above steps constitutes one round of communication, and it continues until the global model converges. FedAvg steps are pictorially shown in **Figure 1**, where steps 1 to 4 form one round of communication. Relevant mathematics is shown in subsection 2.2, which is directly extracted from [8].

## 2.2 Mathematical description

For a distributed system with  $K$  clients, and  $P_k$  being the set of indices of the data points on client  $k$ . If the data generated in these clients is IID and describe the same

process, the finite-sum objective for the overall system can be written as:

$$f(w) = \sum_{k=1}^K (n_k/n * F_k(w))$$

where  $F_k(w) = 1/n_k \sum_{i \in P_k} f_i(w)$  (1)

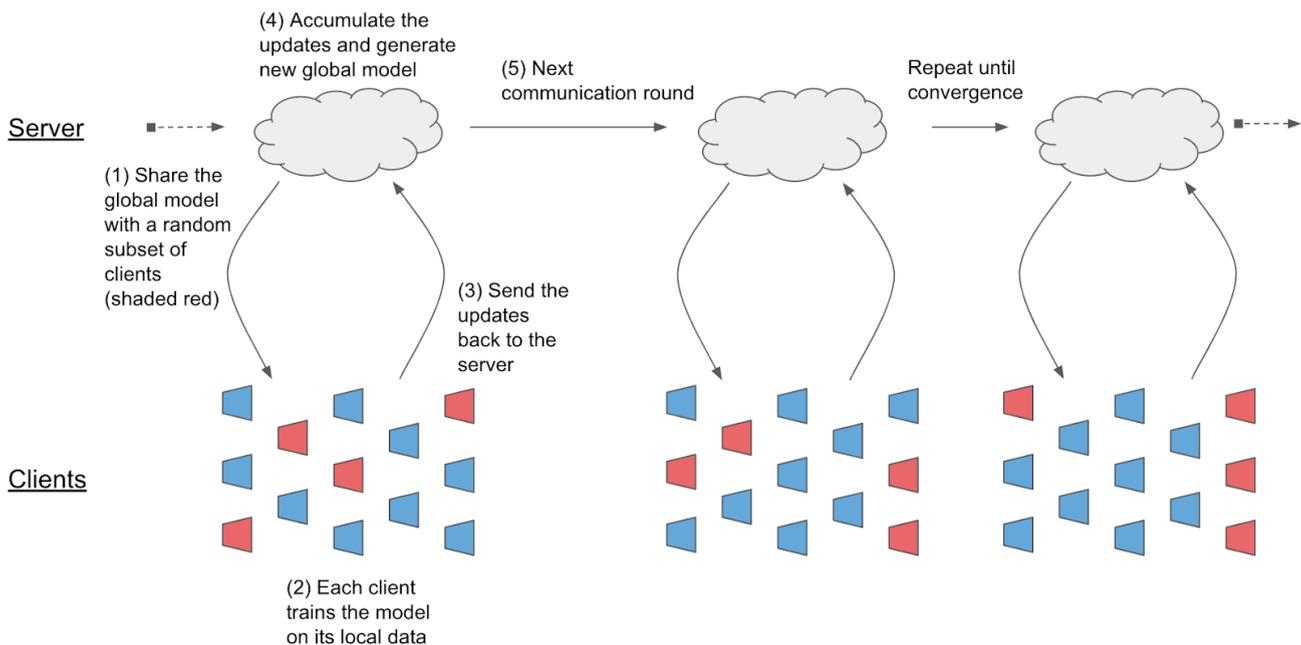
Where  $n_k = |P_k|$  is the total number of data points on client  $k$ . Neural Networks commonly rely on Stochastic Gradient Descent to optimize the objective/ loss. FedAvg extends this for optimizing objective (1), for a fixed learning rate  $\eta$ .

Consider a subset of clients, of size  $C$ , selected in a given round of communication. This subset of clients is called a federation. Each of the clients computes the average gradient  $g_k = \nabla F_k(w_t)$  on its data for the current model parameters  $w_t$ . The server then aggregates the updates from each of the clients, and updates the global model as:

$$w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^K (n_k/n * g_k),$$

since  $\sum_{k=1}^K (n_k/n * g_k) = \nabla f(w_t)$  (2)

The averaging of model parameters of the commonly initialised neural network models described in (2) is empirically shown to converge. Thus, the model benefits from the data in each client, and the weight of its update is proportional to the amount of data locally present in the client. This is the working principle of FedAvg.



**Figure 1.** Pictorial description of FedAvg

### 3. Simulation setup

A fleet of 100 turbofan engines is simulated using the “train\_FD001” part of the publicly available Commercial Modular Aero-propulsion System Simulation (C-MAPSS) dataset [10]. The data for each asset is a run-to-failure data consisting of 24 features (3 representing the operating conditions, and the rest sensors) corresponding to the state of the asset at every timestep. We call the run-to-failure data from an asset’s healthy state till its failure a “trajectory”. All 100 trajectories in train\_FD001 correspond to the same failure type (High Pressure Compressor degradation) and operating conditions. A single asset is simulated using each failure trajectory in the dataset.

The trajectories are multivariate time-series of different lengths. Some of the sensors however do not show any trends throughout. It can be assumed that these sensors do not reflect the asset’s failure behaviour and therefore can be ignored to improve the training process. Concretely, we have ignored the sensors which show standard deviations of less than 0.003 (after normalising the individual sensor values from 0 to 1) over the entire trajectory. Moreover, the noise associated with the sensor values is random. Such noise can be filtered using rolling average (rolling average with window size 40 in our case). In summary, the trajectories are cleaned, the values normalised from 0 to 1, and the relevant sensors selected. After this preprocessing, we get a training dataset consisting of 100 run-to-failures, with 17 features corresponding to each time-step. A representation of the final dataset is shown in **Table 1**.

TensorFlow Federated (TFF) framework [9] is used to implement prognosis algorithm for our simulated fleet. This framework is an architecture similar to [11] used for collaborative prognosis. To draw an analogy, TFF represents “Digital Twins” as clients, and the “Social Platform” as server. Both the server and the clients have three elements each- data repository, the analytics engine, and communications manager. These serve the purposes of storing the locally required data, analysing the local data, and sharing updates with the system respectively. This is shown in **Figure 2**, where the three components are represented using the letters D, A, and C. Detailed information about the role played by these components can be found in [12]. Each asset in our fleet is simulated and represented by a single instance of client. Keras library is used to train the neural networks.

We demonstrate training a feed-forward, and a recurrent neural network (RNN, with one LSTM layer) using FedAvg. We feel that Neural networks, and RNNs in particular, are a good choice for prognosis. This is because of their flexibility in estimating the temporal relation between the features, and corresponding RUL of the asset [13]. The architecture for the RNN we use is  $10*20*50*30*5*1$  with the 10-neuron layer being the LSTM layer. The LSTM layer is removed for the feed-forward network.

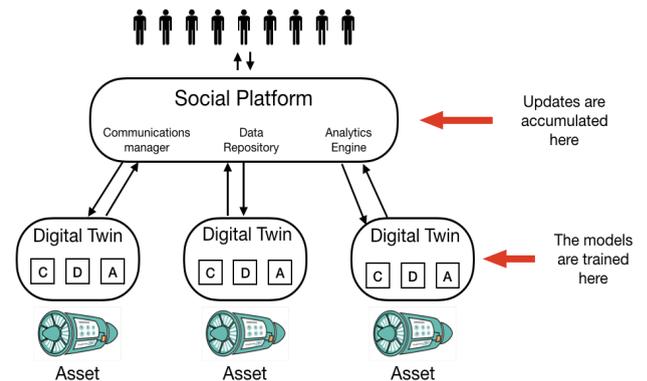
Drawing parallels with **Section 2.2**, the local updates in our simulation are calculated by the clients. That is, the

subset of assets selected at each communication round evaluate the updates for the current neural network weights using gradient descent. Following which, the server aggregates these updates according to equation (2).

We study the effect of different learning rates, loss functions, and the federation size ( $C$ ) on the overall training process.

**Table 1.** The dataset used for simulating the fleet. This is after preprocessing the original FD\_001 dataset.

Asset ID	Timestep	Feat1	Feat2	...	Feat17
1	1	0.4869	0.5437	...	0.6739
1	2	0.4836	0.5625	...	0.6721
....	....	....	....	...	....
2	1	0.5248	0.5625	...	0.6711
....	....	....	....	...	....
2	287	0.4902	0.5708	...	0.3317
...100	... 200	0.4744	0.4562	...	0.3251



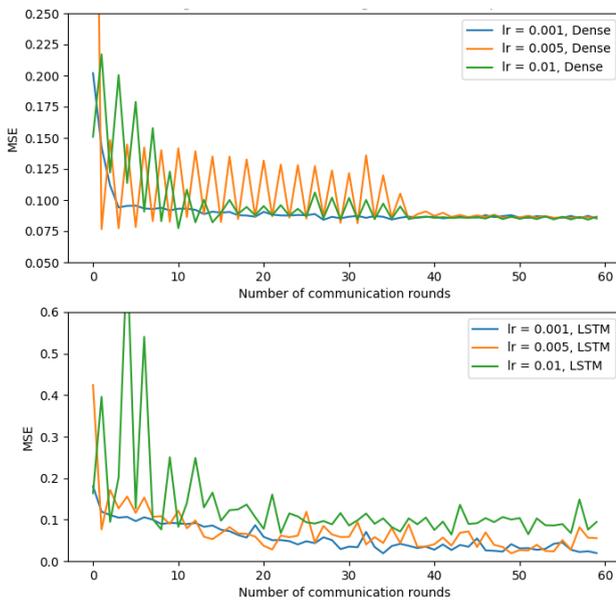
**Figure 2.** The TensorFlow Federated framework

### 4. Results

Out of 100 trajectories, we used 90 for training, and the remaining 10 for testing. This means that the assets reserved for testing the model were never selected in any communication round. Each asset was assumed to have failed once, and therefore the trajectories are evenly distributed over 90 assets. For each set of experiment described in the previous section, our system was run for 60 rounds of communication. A random federation (of fixed size  $C$ ) of assets was selected for each round of communication. The neural network model was trained over these assets, and the trained models were averaged at the server. This averaged model was then passed to the next federation of assets.

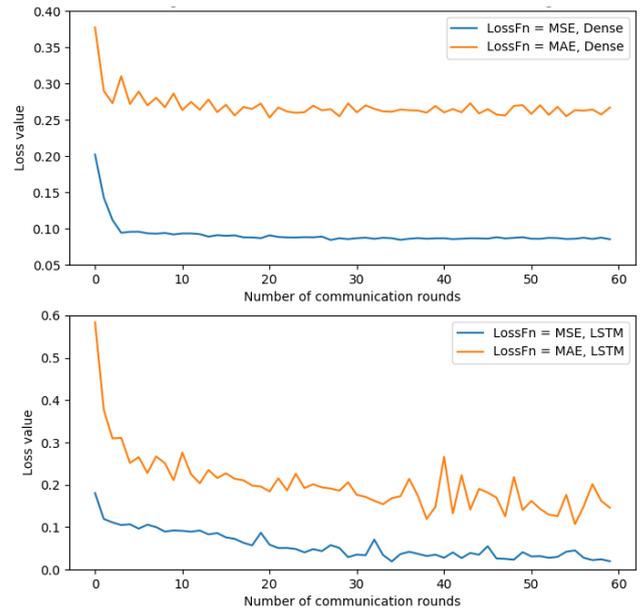
We performed three sets of experiments to analyse the dependance of FL over different learning hyperparameters. In all the experiments we used the Adam optimiser to train the neural networks at the clients level. The results are discussed below:

1. The first set dealt with analysing the effect of the learning rate (lr) on the convergence of global model. This is the learning rate for the gradient descent updates calculated by the clients. The value of  $C$  was fixed at 5. Three different learning rates (0.001, 0.005, 0.01) shown in **Figure 3** were tested to minimise the mean squared error (MSE) between the predicted and real times to failures. The values of loss function after each communication round for first experiment set are shown above. The top graph corresponds to the learning behaviour of feed-forward network, and the bottom one to the LSTM. It is inferred from the plots that for  $lr = 0.01$ , the convergence is erratic whereas it is the smoothest for 0.001. This is because of the fact that a larger learning rate causes the optimiser to oscillate more around the global optima before it finally converges.



**Figure 3.** Effect of learning rate (lr) on the convergence of Mean Squared Error

2. In the second set of experiments, the effect of the loss function on the convergence was analysed. Concretely, we used MSE, and mean absolute error (MAE) between the real and the predicted times to failures. The value of  $C$  was fixed at 5. The results are summarised in **Figure 4**. It is observed that both RNN and feed-forward models converge for mean squared (MSE), and mean absolute error (MAE) as the loss functions. Although the convergence of MSE is smoother.



**Figure 4.** Convergence of different loss functions for 0.001 learning rate

3. The effect of varying federation size ( $C$ ) on the convergence behaviour was analysed in the final (third) set of experiments. We used 0.001 learning rate, and MSE loss function. The choice follows from the previous sets where we saw that this combination shows the relatively smoothest convergence. The values of  $C$  used here were 5, 10, and 20. Similar plot for this experiment set is shown in **Figure 5**. It is inferred that the federation size does not have a significant effect on the convergence behaviour, but the rate of convergence slightly increases as the federation size increases. The initial dip in the loss function values might be because of the random initialisation of the global model parameters, and exploration of the parameter space around that location. However, the values towards the later communication rounds represent the state where the model parameters have stabilised.

To further compare and gauge the performance of FL, we trained an RNN model with the same architecture and test-train split as the one used for FL on the whole dataset together (this is similar to centralised training). The model was trained for 500 epochs (so that the model had converged to its near-optimal parameter values), and a batch-size of 5. The FL model trained with  $C = 20$  and Adam optimiser with learning rate 0.001 was used for comparison. This was the best convergence we had achieved from our experiments. The results of this comparison are shown in **Figure 6**. Quantitative comparison was made using the penalty metrics advised in [10] for the dataset we used with slight modification. In our calculations, the trajectories were divided into three equal parts. These sections were weighted in 1:2:3 ratio

while calculating the scores. This is because the accuracy of prediction becomes increasingly important as we approach the point of failure. Centralised and FL trained models were scored a penalty of 5.92 and 118.85 respectively.

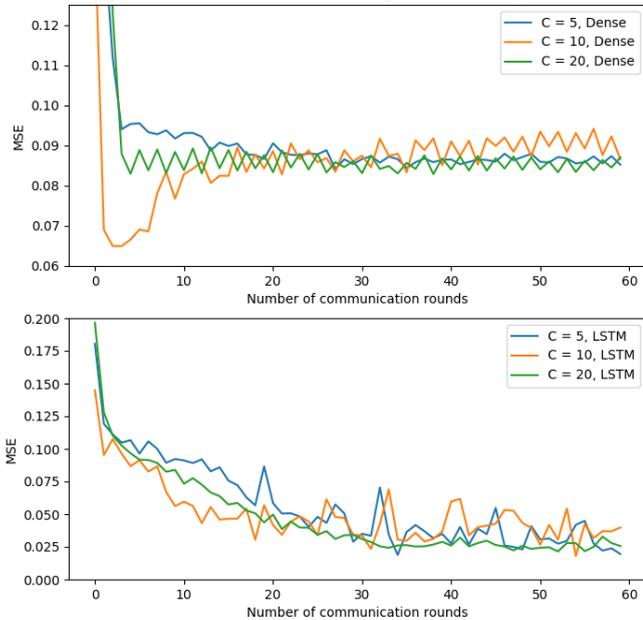


Figure 5. Effect of Federation size (C) on the convergence of Mean Squared Error (MSE)

### 5. Conclusions

It can be concluded from the results discussed above that FL is applicable for fleet prognosis, and that it is a promising path to follow for realising the implementation of collaborative prognosis. This follows from the observation that all the plots in Figures 3 to 5 show convergence (although slow and erratic for some) of the global neural network model. Figure 6 also pictorially shows that the predictions of FL model tend to follow the real values. The various steps for the algorithm when implemented for a fleet of assets are shown as a flowchart in Figure 7 for a clearer representation. These steps are

followed after similar assets have been clustered together. FedAvg enables these similar assets to learn from one another without explicitly exchanging data.

For the experiments discussed here, the training data was evenly distributed across the entire fleet. The assets were all similar, and therefore the fleet simulated here does not reflect the real world where the assets can be widely different. But the important aspect is the global model in our experiments was trained without moving the data away from the assets, or exchanging data within different elements of the computing framework. Moreover, the server also did not access the data at any of the clients (assets) for training the model. This is in contrast to collaborative prognosis where the data is extensively shared within the assets and the server, thus limiting its practical implementation.

### 6. Future Work

We have identified the need for moving towards model-based (Federated) learning for assets fleet prognosis. Moreover, the work presented in this paper is the earliest implementation of Federated Learning (FL) for fleet prognosis.

To further this work, custom loss functions can be used to improve predictions. Additionally, this work can be extended to heterogenous networks- like collaborative prognosis where similar assets are clustered together before they learn from one another. FL algorithms can be developed for fleet prognosis so that the dissimilar assets can appreciate their differences, and modify their models as they learn from each other.

### Acknowledgement

This research was partly supported by Siemens Industrial Turbomachinery UK. The project that has generated these results has been supported by a “la Caixa Fellowship (ID 100010434), with code LCF/BQ/EU17/11590. This research was supported by RAE project IAPP 18-19/31.

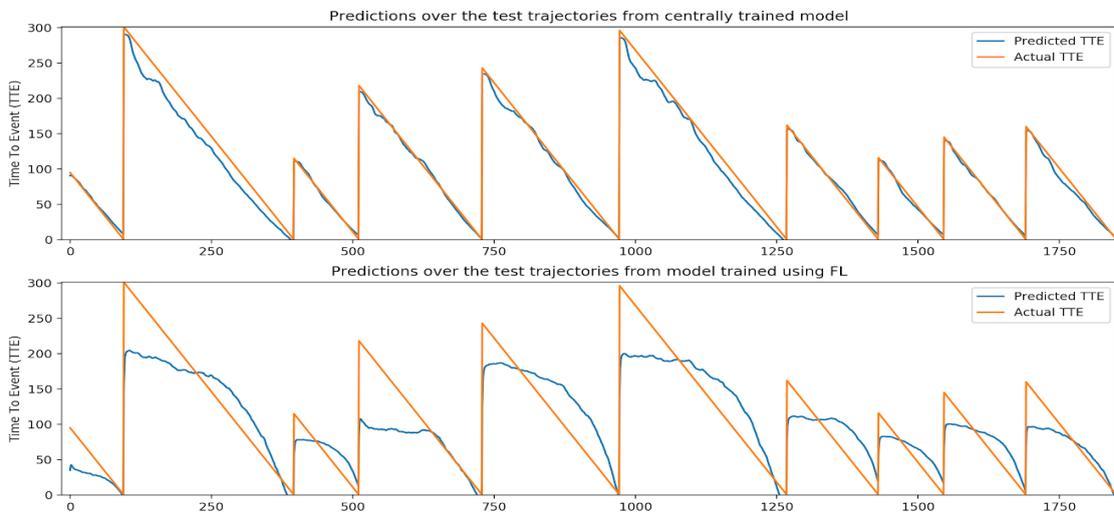


Figure 6. Predictions of centrally vs FL trained models

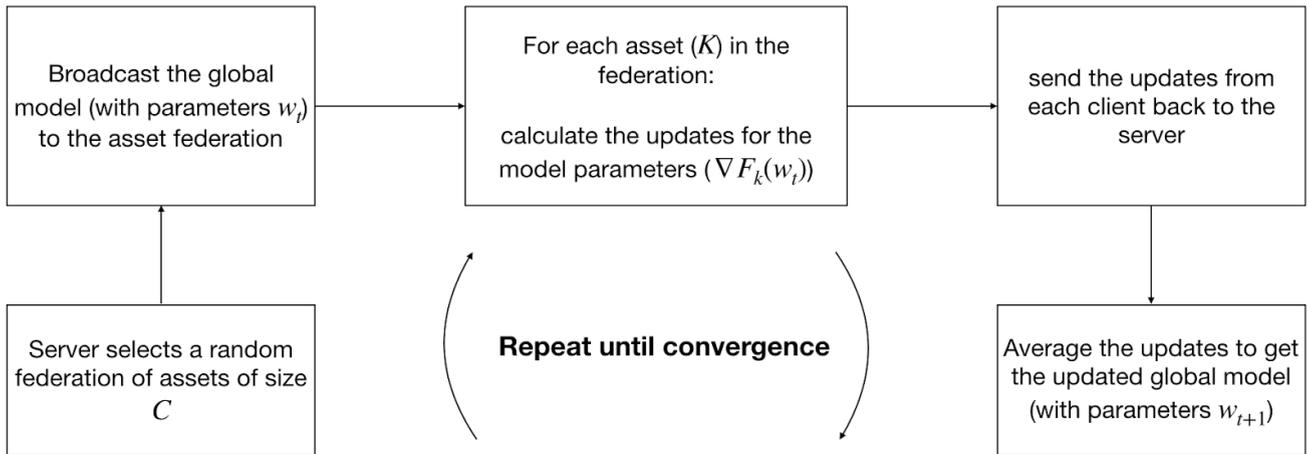


Figure 7. Flowchart above describes the FedAvg algorithm steps at each communication round.

## References

- [1] McFarlane D (2018) Industrial Internet of Things: Applying IoT in the Industrial Context
- [2] Li, H., Palau, A.S. and Parlikad, A.K., 2018. A social network of collaborating industrial assets. Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability, 232(4), pp.389-400.
- [3] Palau, A.S., Liang, Z., Lütgehetmann, D. and Parlikad, A.K., 2019. Collaborative prognostics in social asset networks. Future Generation Computer Systems, 92, pp.987-995.
- [4] Siemieniuch, C.E. and Sinclair, M.A., 2002. On complexity, process ownership and organisational learning in manufacturing organisations, from an ergonomics perspective. Applied Ergonomics, 33(5), pp.449-462.
- [5] Palau, A.S., Dhada, M.H. and Parlikad, A.K., 2019. Multi-agent system architectures for collaborative prognostics. Journal of Intelligent Manufacturing, pp.1-15.
- [6] Lian, X., Zhang, C., Zhang, H., Hsieh, C.J., Zhang, W. and Liu, J., 2017. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In Advances in Neural Information Processing Systems (pp. 5330-5340).
- [7] Konečný, J., McMahan, B. and Ramage, D., 2015. Federated optimization: Distributed optimization beyond the datacenter. arXiv preprint arXiv:1511.03575.
- [8] McMahan, H.B., Moore, E., Ramage, D. and Hampson, S., 2016. Communication-efficient learning of deep networks from decentralized data. arXiv preprint arXiv:1602.05629.
- [9] Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konecny, J., Mazzocchi, S., McMahan, H.B. and Van Overveldt, T., 2019. Towards Federated Learning at Scale: System Design. arXiv preprint arXiv:1902.01046.
- [10] Saxena, A., Goebel, K., Simon, D. and Eklund, N., 2008, October. Damage propagation modeling for aircraft engine run-to-failure simulation. In 2008 international conference on prognostics and health management (pp. 1-9). IEEE.
- [11] Palau, A.S., Dhada, M.H., Bakliwal, K. and Parlikad, A.K., 2019. An Industrial Multi Agent System for real-time distributed collaborative prognostics. Engineering Applications of Artificial Intelligence, 85, pp.590-606.
- [12] Bakliwal, K., Dhada, M.H., Palau, A.S., Parlikad, A.K. and Lad, B.K., 2018. A Multi Agent System architecture to implement Collaborative Learning for social industrial assets. IFAC-PapersOnline, 51(11), pp.1237-1242.
- [13] Palau, A.S., Bakliwal, K., Dhada, M.H., Pearce, T. and Parlikad, A.K., 2018, June. Recurrent Neural Networks for real-time

distributed collaborative prognostics. In 2018 IEEE International Conference on Prognostics and Health Management (ICPHM) (pp. 1-8). IEEE.