IET Collaborative Intelligent Manufacturing

Secure and Communications Efficient Collaborative Prognosis

CIM-2020-0035 | Special Issue: Smart Manufacturing (including selected papers from 11th COPEN conference)

Submitted by: Maharshi Dhada, Amit Kumar Jain, Manuel Herrera, Marco Perez Hernandez, Ajith Kumar Parlikad

Keywords: SMART MANUFACTURING, COLLABORATIVE INTELLIGENT MANUFACTURING, MACHINE LEARNING, FAULT PROGNOSTICS

IET Research Journals

Research Article

IET Journals
The Institution of Engineering and Technology

ISSN 1751-8644
doi: 0000000000
www.ietdl.org

Secure and Communications Efficient Collaborative Prognosis

Maharshi Dhada[1*], *Amit Kumar Jain*[1], *Manuel Herrera*[1], *Marco Perez Hernandez*[1], *Ajith Kumar Parlikad*[1]

[1] *Institute for Manufacturing, Department of Engineering, University of Cambridge, Cambridge, U.K. CB3 0FS*
* *E-mail: mhd37@cam.ac.uk*

Abstract: Collaborative prognosis is a technique that is used to enable assets to improve their ability to predict failures by learning from the failures of similar other assets. This is typically made possible by enabling the assets to communicate with each other. The key enabler of current collaborative prognosis techniques is that they require assets to share their sensor data and failure information between each other, which might be a major constraint due to commercial sensitivities, especially when the assets belong to different companies. This paper uses Federated Learning to address this issue, and examines whether this technique will enable collaborative prognosis while ensuring sensitive operational data is not shared between organisational boundaries. An example implementation is demonstrated for prognosis of a simulated turbofan fleet, where Federated Averaging algorithm is used as an alternative for the data exchange step. Its performance is compared with conventional collaborative prognosis that involves failure data exchange. The results confirm that Federated Averaging retains the performance of conventional collaborative prognosis, while eliminating the exchange of failure data within assets. This removes a critical hinderance in industrial adoption of collaborative prognosis, thus enhancing the potential of predictive maintenance.

1 Introduction

Advances in sensor, communication, and computing technologies over the past decades have propelled extensive automation of the industrial systems [1]. Manufacturing industries have also moved towards servitisation, where the customers pay for the services rather than the assets. The original equipment manufacturers therefore need to bear the associated costs for asset upkeep and maintenance [2].

Industrial automation has been amongst the key enablers for servitisation. As a result of the technological advances, the industries are capable of monitoring their assets in real time via embedded sensors [1]. The sensor data enables the operators to closely monitor an asset's health and implement state-of-the-art predictive maintenance strategies, based on the asset's predicted remaining useful life. An asset's remaining useful life refers to the remaining time before an impending failure, after which the asset would be deemed not capable of operating satisfactorily [3].

Prognosis, or prediction of impending failures, particularly has moved from traditional physics based formulations to data driven techniques [3, 4]. As a critical precursor to the modern maintenance planning strategies, accurate prognosis can significantly boost the efficiency of an industrial system [5, 6]. Data driven prognosis involves Machine Learning (ML) techniques to learn a failure prediction model using historical failure data. This model is then expected to predict similar impending failures in real time. Data driven prognosis is advantageous for those failure types whose mathematical formulations based on the physical failure laws are not straightforward [3].

Primary sources of failure data are the sensors embedded at various internal locations across an industrial asset. Measurements recorded by these sensors over a period constitute time series data indicating the asset health at corresponding instances. Time series data ranging from an asset's healthy condition until its failure is called a failure trajectory. ML algorithms rely heavily on historical failure trajectories to train a prediction model for that failure type [3]. The analytics pipeline for data driven prognosis involves (1) identifying a failure type for given operating conditions of assets, (2) training prediction model using historical trajectories of that failure, and (3) implementing the trained prediction model in real time [3, 5].

However assets, especially those with high reliability, might not posses sufficient failure trajectories necessary for training a prediction model. [7].

In this context, collaborative prognosis is a technique that enables a network of assets, comprising a fleet, to learn from one another [7–11]. It involves identifying clusters of assets that operate in similar conditions and have encountered same failures, followed by sharing failure trajectories within these asset clusters. As a result, any given asset's data repository is enriched with failure trajectories originating from other assets. Prediction models are then trained using the enriched dataset [9]. Other such similarity-based prognosis have also been proposed by researchers [12–14].

While in early 2000s prediction models were trained in remote cloud servers, Internet of Things and increasing power of edge computing resources have enabled localising data analytics at the asset level. Advantages of such distributed computing frameworks for industrial systems can be found in [15, 16]. Several distributed system architectures and protocols have also been postulated for various industrial systems [16–18]. As such, authors believe that infrastructural support and benefits of distributed data driven prognosis techniques, like collaborative prognosis, are sufficiently present. However, certain practical challenges hinder their practical implementation.

This paper targets challenges caused specifically due to sharing failure trajectories across assets. While collaborative prognosis is lucrative for the original equipment manufacturer, it is risky from an operator's perspective. This is because many real world operators would not want their asset data to be shared with their competitors [19, 20]. Exchanging failure trajectories also increases avoidable network communication costs [11]. Such practical challenges hinder practical implementation of collaborative prognosis in industry.

Federated Learning [21] is proposed in this paper as a solution to address above described challenges. Federated learning methods aim at shifting model training to nodes of a networked system. It has gained immense popularity across various applications in recent years due to rise of awareness about data privacy [22, 23]. Federated Averaging (FedAvg) is a primitive and widely analysed Federated Learning algorithm for domains like healthcare, mobile devices, home security systems, etc [23]. This paper describes the application of FedAvg for training recurrent neural networks for prognosis of failures in a fleet of industrial assets.

The outline for the rest of the paper is as follows: Section 2 discusses collaborative prognosis, state-of-the-art literature in Federated Learning, and FedAvg in the context of asset prognosis. In order to demonstrate an example application of FedAvg for asset prognosis, simulated failure trajectories were used to replicate real world fleets with failures distributed across several assets. The simulated dataset and underlying computational framework used for experiments are explained in Section 3. Section 4 describes the experiment cases that were conducted as a part of analysis. Experimental results are presented and discussed in Section 5. Lastly, important conclusions and future research directions are summarised in Section 6 and 7 respectively.

2 Background

This section discusses the state-of-the-art research in collaborative prognosis techniques, Federated Learning, and also describes the FedAvg algorithm in the context of prognosis.

2.1 Collaborative Prognosis Techniques

The performance of data driven prognosis relies heavily on the historical failure data used for training the prediction models [24]. For prognosis, just like other machine learning applications, the prediction models tend to learn faster and be more accurate if its training data is statistically homogeneous i.e. Independent and Identically Distributed (IID). IID data refers to those cases where the individual data points can be considered independently sampled from a common underlying probability distribution. In the context of prognosis, IID data refers to same failures occurring in machines operating under similar conditions [3, 7]. However, an industrial system of assets is often characterised by widespread heterogeneity, due to assets operating in varied conditions and presence of multiple failure modes. It has been shown that in such settings, it is beneficial to have separate prediction models catering to subsets of asset populations, identified based on some sense of homogeneity [7, 9].

It has been shown that the failure predictions are most accurate if the model learns from a single asset only [7]. Recently popularised distributed computing architectures for industrial systems enable every asset in the fleet to have its own corresponding prediction model [8, 9, 11, 16]. However, the individualised models would require assets to fail a certain number of times so that necessary training data is available [7, 24]. Collaborative prognosis technique aims at reducing these asset failures, by enabling assets to identify other similar assets in the fleet and learn from their failures as well [7]. This is made possible by identifying *clusters* of similar assets and exchanging failure data across assets comprising these clusters [7, 9]. Collaborative prognosis is most suitable for assets with high reliability, like flight engines, where individual assets would not experience enough failures to generate sufficient training data [10].

Identifying clusters of similar assets/ failures has also been the basis of many data driven prognosis techniques presented in literature. [12] showed that in a system comprising of multiple assets and historical failures, prediction of a given asset is improved by identifying similar historical behaviours from a library of past failure data, and evaluating the best fit for the current failure's degradation curve. [14] used genetic algorithm to identify clusters of the most similar historical failure trajectories, which in turn improved the prediction accuracy of models corresponding to each of those identified clusters. Example implementation of this was shown for fatigue crack growth, drilling bit degradation, and degradation of a turnout system applications. [13] relied on collaborative learning to tackle the lack of sensing resources for the overall cohort of units, for the cases of both medical patients and industrial assets. Collaborative learning in this case was based on Markov models and selective sensing to address the problem of incomplete data per individual units.

The most recent collaborative prognosis implementation involves distributed deployment of all constituting steps, ranging from identification of similar assets, training the models, and real time failure prediction. It has been shown that distributed collaborative prognosis is more adaptable, scalable, resilient, flexible, and lean than the former techniques which were deployed on centralised cloud servers [9].

This paper focuses on distributed collaborative prognosis presented in [9], which involves exchanging failure trajectories across assets comprising the clusters. Failure data exchange step of distributed collaborative prognosis is identified as a major impediment for realising distributed collaborative prognosis in industries.

2.2 Federated Learning

As their computing capabilities improved, it is now possible for the user devices like mobile phones to participate in data analytics and therefore reduce computational burden on a central cloud server. Shifting computation to devices is referred to as edge, or fog computing across diverse applications [23]. For physical industrial assets, embedded microprocessors enhance their digital capabilities and make them "smart" enough to perform analytics locally [15].

However, distributed systems pose different algorithmic requirements than cloud computing. In contrast to cloud computing, computing on end user devices involves increased communications. Communication is key to analytic performance because data are stored at distant nodes across the system. Ideally computing on a network of nodes is equivalent to computing on multiple processors housed in a single server. But inefficiencies of the network connections, differences in the technical capabilities of individual nodes, and statistical heterogeneity of data across nodes cause synchronisation issues, presence of straggler and dropout nodes, and several other data handling related issues [25]. Distributed optimisation challenges can be summarised as (1) expensive communication, (2) systems heterogeneity, (3) statistical heterogeneity, and (4) data security [23]. The majority of research in distributed ML is focused on achieving improved model performance in the presence of these challenges.

Federated Learning (FL) refers to those learning techniques which focus their application specifically for distributed systems where the communication costs and the data security hold prime importance. It is called "Federated" because only a federation of network nodes participate in the learning process at a given instance. Target applications of FL are characterised by local computations being orders of magnitudes faster than communications due to network size, or where data must not leave the nodes [21]. Both these constraints hold for asset prognosis [11, 19].

FL involves storing and processing the data at its origin, and sharing only certain updates with a central server. FL methods have been deployed by major service providers and proposed as a critical enabler of several data-sensitive applications including [26–28].

Basic FL problem formulation involves learning a *single global statistical model* representing data stored across the nodes. This model is learnt by optimising a global objective function for the entire network, which in turn involves jointly optimising local objective functions at the nodes [21]. The local objective functions might as well be different from the global objective function. The global objective function $F(w)$ for a network of m nodes is mathematically represented in (1). Here, the local objective functions are denoted by F_k for k^{th} node, with w being their corresponding model parameters. p_k is weight associated with the k^{th} node. Choice of p_k varies across applications but popular choices are $p_k = \frac{n_k}{n}$ or $p_k = \frac{1}{m}$, where n_k is the data at k^{th} node and n is the total data across the entire network.

$$\min_w F(w), \qquad \text{where} \qquad F(w) := \sum_{k=1}^{m} p_k F_k(w) \tag{1}$$

While (1) is the generic mathematical formulation, FL literature has instances where multiple objective functions have been proposed for catering to underlying statistical heterogeneities [29].

FedAvg is an FL technique which enables learning a single Artificial Neural Network (ANN) model for data distributed across network nodes. The reader must not be confused with two usages of "network", which in its former instance in previous statement refers to the network of neurons constituting the ANN, while in the later

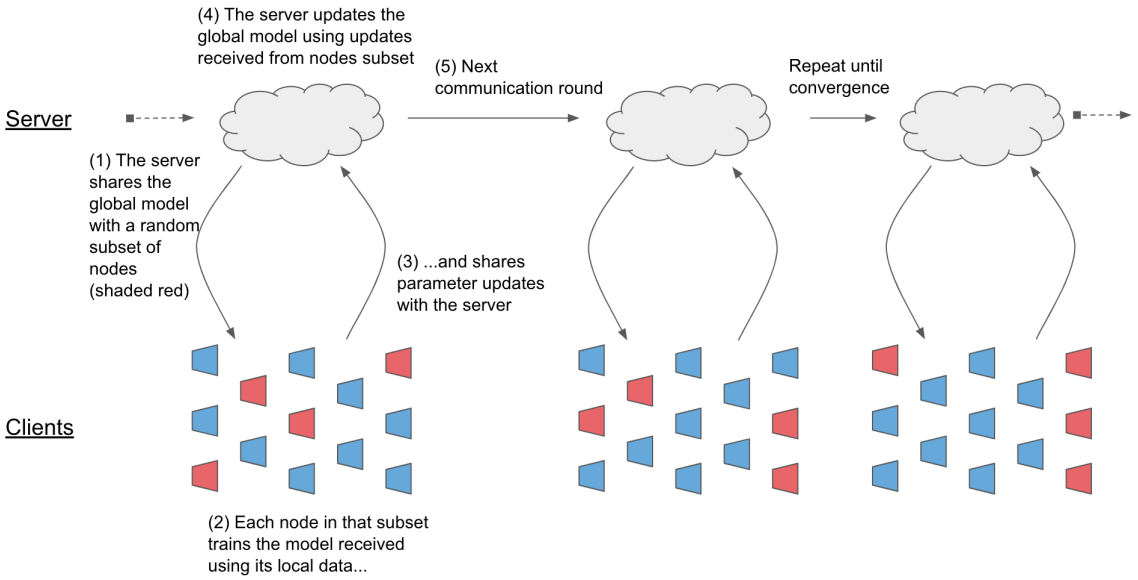

Fig. 1: Schematic representation of steps involved in a single communication round of FedAvg.

usage it refers to the physical network of computing nodes. ANNs are a family of ML techniques, which are based on underlying principle similar to biological brains. More information about ANNs can be found in [30].

2.3 FedAvg in the Context of Asset Prognosis

Federated Averaging (FedAvg) is a primitive, and amongst widely analysed FL methods, especially suited for training ANNs for data distributed across nodes. FedAvg enables the network nodes to train a global ANN model using their individual data, and share only the parameters of the trained model with the server. The server accumulates parameters from all participating nodes and updates the global model which, after complete training, represents the general statistical behaviour of data across nodes [21].

The loss surfaces of sufficiently over-parameterised artificial ANNs are well behaved and escape bad local minima. Therefore, when two ANN models with the same parameter initialisations are trained independently on different subsets of IID data, naive averaging of their updated parameters can be used to obtain a single model describing combined data. This is the underlying principle of FedAvg [31]. The performance for averaged model in some cases can also be better than either of the two models [31]. However, over-parameterising the ANNs also leads to increased need for training data and overfitting hazard. Therefore, the ANN architecture must be carefully analysed by the users, and the number of parameters must be kept at the bare minimum necessary for FedAvg. Further information about the effect of ANN parameters on its training can be found in [30].

FedAvg is applicable where the ANN models are trained using gradient descent methods and for statistically homogeneous (IID) datasets only [29]. In FedAvg, a random subset of network nodes parallely updates global model parameters based on their data using a gradient descent method. The updated model parameters from these nodes are averaged by the server to obtain a new, updated, global model. Often, if the data are non-uniformly distributed across nodes, weighted averaging is used to aggregate parameter updates at the server. After this, the updated global model is again shared with a new randomly selected subset of nodes, and the same process repeats. A single communication round comprises of local updates at the nodes followed by parameter aggregation at the server. After several such communication rounds, the global model converges and describes cumulative data across all nodes [21]. A schematic representation of the above described steps is shown in Figure 1.

For the case of asset prognosis, training data comprises of historical failure trajectories, which are distributed across several assets. Assets lie at the nodes of the network, that could have varying instances of failure occurrences. Since FedAvg requires data to be IID, each failure type has a prediction model specifically trained for its prediction. The clustering step in collaborative prognosis helps identifying such clusters comprising IID failure trajectories, as discussed in Section 2.1. FedAvg is proposed in this paper as the step following the clustering step in collaborative prognosis, to train prediction models for each of these identified homogeneous clusters. The parameters involved and mathematical description of FedAvg for asset prognosis are presented in the following subsection.

2.4 Mathematical Description

The standard mathematical description of FedAvg is presented here, and its application for asset fleet prognosis is explained based on this description.

Let us consider a distributed system comprising of m nodes, total data across all nodes be n, and n_k be amount of data at node k. Of these m nodes, consider a subset of nodes having size S_t be selected at the t^{th} communication round. This subset of nodes is called a federation, which is generally expressed as a fraction C of nodes selected from the total m nodes, such that $|S_t| = \max(int(C * m), 1)$. Where $int(C * m)$ means the highest integer less than or equal to $(C * m)$ Parameter C influences the model performance and also the overall learning process, and therefore must be carefully selected depending on the application. The effects of the parameters governing the FedAvg learning process are discussed in Section 5.

Fraction C is constant for every communication round. Each node in S_t computes average gradient of local objective function $F_k()$, for current parameters of the global model and using its data. This gradient is given by $g_k = \nabla F_k(w_t)$, where w_t are global model parameters at t^{th} communication round. The server then generates the global model for the next round as:

$$w_{t+1} \leftarrow w_t - \sum_{k \in S_t} \left(\frac{n_k}{f_{S_t}} * g_k \right), \quad \text{since} \quad \sum_{k \in S_t} \left(\frac{n_k}{f_{S_t}} * g_k \right) = \nabla F(w_t) \qquad (2)$$

Algorithm 1: Steps followed while implementing FedAvg for a fleet containing n failure instances across m assets.

Result: To optimise the global objective function $F(w)$, explained in (1) using local updates explained in (2)

At the Server:
initialise w_0;
for *each communication round t = 1,2,3...* **do**
  $s \leftarrow \max\big(int(C * m), 1\big)$;
  $S_t \leftarrow$ (random subset of s assets) ;
  $f_{S_t} \leftarrow \sum_{k \in S_t} n_k$;
  for *each asset* $k \in S_t$ ***in parallel*** **do**
    $w_{t+1}^k \leftarrow \text{AssetUpdate}(k, w_t)$;
  end
  $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{f_{S_t}} w_{t+1}^k$;
end

AsssetUpdate(k, w):
for *each local epoch i from 1 to* E **do**
  $w \leftarrow w - \lambda \nabla F_k(w, n_k)$;
end
return w to the server

Where f_{S_t} are total failures in S_t subset of assets. Local updates are further governed by their corresponding ANN parameters including epochs per node (E), the optimiser, and the learning rate (λ) of the optimiser used for training [31]. Overall, FedAvg Algorithm is governed by three main parameters: (C, E, λ). Batch size corresponding to local training of ANNs can also be varied, but it is considered for prognosis applications that a single asset does not fail often and therefore the batch size for local updates would not substantially effect the learning process.

Several distributed computing architectures exist that enable collaborative prognosis in physical assets with computing capabilities, and therefore also make them capable for FedAvg. One such architecture used for experiments discussed in this paper is explained in Section 3. FedAvg steps are summarised in Algorithm 1, which adapted from [31].

While applying for asset prognosis, parameter m introduced above corresponds to total assets included in a given cluster, n to total number of instances of that failure across all comprising assets, and n_k to its number of instances at asset k.

Conventional collaborative prognosis involves exchanging failure trajectories amongst all participating assets. For a cluster comprising total n failures, m assets, and b_{trajec} being the size of single failure trajectory data, $[(m).(m-1).(n).(b_{trajec})]$ amount of data would need to be transmitted across the network during the training process. On the other hand, FedAvg involves only sharing model parameters between assets and the server. Therefore, a total of $[(m).(C).(b_{model}).(r)]$ data would be transmitted across the network, where b_{model} is the size of model parameters data and r are total communication rounds. For most prognosis applications, failure trajectory data is significantly higher than model parameters data [11]. Moreover, transmitted data exponentially increases with increasing number of assets for the case of conventional collaborative prognosis. Data transmission for FedAvg on the other hand linearly increases with increasing number of assets, and is independent of total number of failures in the fleet.

3 Implementing FedAvg for Prognosis

This section describes the dataset and the enabling architecture for collaborative prognosis that were used for the experiments.

3.1 Dataset Description

Dataset used for experiments discussed here is the publicly available Turbofan Engine Degradation Simulation Data Set [32]. This dataset is generated using a Matlab based simulator called Commercial Modular Aero-Propulsion System Simulation (C-MAPSS), and therefore will subsequently be referred to as the C-MAPSS dataset. Detailed description of its simulator can be found in [33].

Table 1 A Sample of FD_001 Dataset

AssetID	Cycles	OC1	OC2	OC3	s1	...	s21
1	1	-0.0007	-0.0004	100	519	...	23.419
1	2	0.0019	-0.0003	100	519	...	23.424
...	...	...	...	...	...	...	...
2	1	-0.0018	0.0006	100	519	...	23.458
...	...	...	...	...	...	...	...
2	287	-0.0005	0.0006	100	519	...	23.084
...	...	...	...	...	...	...	...
100	200	-0.0032	-0.0005	100	519	...	23.052

C-MAPSS is capable of simulating turbofan engines operating under various user defined operating conditions. These conditions include the altitude at which the engine is operating, its Mach number, and the temperature at sea-level conditions. Thermodynamic equations are used to calculate fluid flow parameters, and health conditions of engines are reflected in sensor measurements from various internal locations. A single simulated turbofan is monitored using 21 sensors [33].

Turbofans also comprise of independent sub-systems including regulators, limiters and control systems. The limiters resemble warning-trip mechanisms typically present in industrial turbomachinery that prevent machines from exceeding pre-set tolerances. In C-MAPSS, there are limiters for the core speed, the engine-pressure ratio, for the high pressure turbine exit temperature, and for the static temperature at the high-pressure compressor. An engine is deemed inoperable/ failed when any of the limiters are exceeded [33].

The C-MAPSS dataset represents several simulated turbofans with continuously degrading health, until they eventually fail. A turbofan's degradation is manifested in simulations as percentage reduction in a component's efficiency($e(t)$) and flow($f(t)$) values at time step (t) compared to those at its healthy state (at time step $t = 0$). The overall health index of a machine at time t is a combined function of flow and efficiency of the overall engine: $H(t) = g(f(t), e(t))$.

The $e(t)$ and $f(t)$ values of a given component are simulated to degrade with time according to an inverse exponentially decreasing function. However, no two simulated turbofans would be identical because the parameters governing inverse exponential function are randomly chosen from their permissible range of values. Turbofans also commence operation with a slight but random initial deterioration to replicate real world manufacturing inefficiencies, and noise is added to the sensor measurements to replicate real world errors [33].

As a result of a turbofan's health degradation, the fluid flow parameters recorded by sensors across various components deviate and trend away from their normal operation values. Time series of sensor measurements ranging from a given turbofan's healthy state until its failure are saved with their corresponding timestamps and operating conditions. These failure trajectories are analogous to real world trajectories used to train prediction models. A sample of C-MAPSS data is shown in Table 1, where the columns indicate unit id, cycles (or timestamp of measurement), operating conditions, and sensor measurements with their corresponding sensor tags. The data shown in Table 1 is sampled from the FD_001 file, which is explained in the following paragraphs.

The C-MAPSS dataset is divided into four files, each of them comprising of failure trajectories for simulated turbofans operating in various conditions and incipient failure modes. Files FD_001 and FD_003 specifically comprise of data corresponding to simulated degrading turbofans operating at sea-level conditions only, and are used for conducting experiments. All turbofans represented in

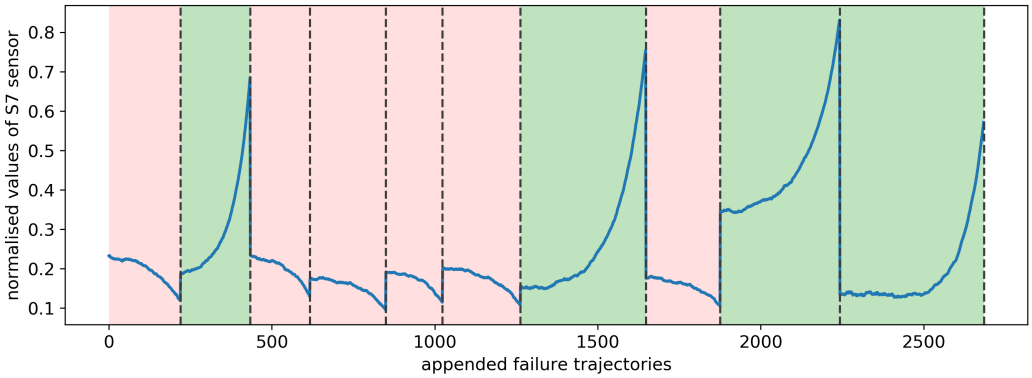

Fig. 2: Shown above are sample rolling mean averages of normalised sensor 7 measurements, that trend differently for high-pressure compressor degradation and fan degradation failure modes in FD_003 turbofans. Identification of failure modes is similar to [9]

FD_001 were simulated to fail because of their high-pressure compressor degradation, and turbofans in FD_003 could fail either due to high-pressure compressor degradation or fan degradation.

As explained in Section 2.3, FedAvg is applicable for IID data only [29]. To conform with this requirement, only files FD_001 and FD_003, where the turbofans operate in same conditions throughout, were used for experiments. All trajectories in FD_001 were used for experiments, but only those trajectories in FD_003 corresponding to failures caused by high-pressure compressor degradation were identified and merged with FD_001 dataset. By visually observing the trends in sensor measurements, it was possible to identify the corresponding failure modes for turbofans in FD_003 with 100% accuracy. This classification for a sample of data from FD_003 is shown in Figure 2, where high-pressure compressor degradations amongst concatenated failure trajectories are indicated using pink background colour, and fan degradations with green.

Finally, a single file containing 148 failure trajectories corresponding to simulated turbofan failures, operating at sea level conditions and incipient high-pressure compressor failure, was obtained and used for experiments. This data was further preprocessed before being used for analysis.

Sensors recording consistent measurements were removed for better training. Concretely, sensors corresponding to measurements with standard deviation of less than 0.003 were removed from data file. The cycles column of every trajectory was inverted to obtain remaining useful life (RUL) for the corresponding feature values. RUL column served as the output/ target variable. Furthermore, values of remaining sensors, operating condition indicators, and RULs were scaled using MinMax scaler, so that their values across the entire file ranged from 0 to 1. RULs were scaled because the range of the output neuron in the RNN, used as the prediction model and explained in Section 4.1, was from 0 to 1. After preprocessing, the trajectories comprised of unit ids, scaled cycle numbers, scaled operating condition values, and scaled measurements from 15 sensors. Out of 148 trajectories, 10 trajectories were set aside for testing.

3.2 System Architecture

The multi-agent system architecture presented in [8–10] was used as the underlying architecture for experiments discussed here. This architecture has been shown to be well suited for implementation in real world industries, and is analogous to the nodes-server (or clients-server) network type suitable for deploying federated learning algorithms [9].

Similar to a nodes-server network, where several computing nodes representing user devices are connected to a central server, the architecture used for experiments discussed here involves a network of connected industrial computing agents. It is formally a *modified hierarchical architecture* type, where every asset in the fleet is monitored and controlled by its corresponding agent. Asset agents analyse data and make decisions for their corresponding assets. They are all connected to a central agent which is responsible for higher level decisions making. Concretely, the architecture comprises of three levels: virtual assets, digital twins, and a social platform. The digital twins and the social platform are implemented for experiments discussed here, and are therefore briefly described in the following paragraphs. Detailed description of entire architecture and industrial multi-agent systems in general can be found in [8] and [16, 34, 35] respectively. The notion of "agent" in this paper refers to a collection of computational entities, that cooperate or compete to achieve a certain objective [36].

Digital Twins: Digital Twin is an asset's local data analyser. It is responsible for monitoring data and extracting operationally useful information. Apart from analysing the data, digital twins can also serve as local decision makers. However, digital twins in the experiments discussed here only act as data analysers for prognosis. Operational decisions for mitigating impending failures are governed by operator policies and asset criticality, and are therefore not discussed here.

A Digital Twin is segmented into data repository, analytics engine, and output manager. The data repository stores data streaming in from other agents, the analytics engine analyses data stored in the repository, and the output manager manages communications between the Digital Twin and agents it is connected with (such as the social platform and the Virtual Asset).

Social Platform: The social platform is a central agent to which all digital twins are connected, and therefore serves as the overall network enabler. It enables communications within the network, and is also responsible for conducting higher level analysis for the overall asset fleet such as identifying clusters of similar assets or registering queries from newly introduced assets. The social platform is also segmented into a data repository, an analytics engine, and a communications manager. The functions served by these are similar to those for digital twins, but the only difference being that the analytics engine of the social platform conducts higher level analysis.

This paper, and experiments discussed herewith, focus only on the model training step of collaborative prognosis pipeline. Therefore, we assume that the participating assets have already been deemed similar by the clustering step, and shown here is the model training step for a given cluster. Moreover, since only historical failures are used to train the prognosis algorithm, implementing virtual assets to standardise online data are deemed not necessary for experiments discussed here. Trajectories selected from the preprocessed dataset explained in Section 3.1 for various experiment cases are stored directly in data repositories of the digital twins to replicate failures distributed across assets constituting a fleet. The only task performed by the analytics engines of the digital twins is to evaluate local updates using trajectories stored in their corresponding repositories.

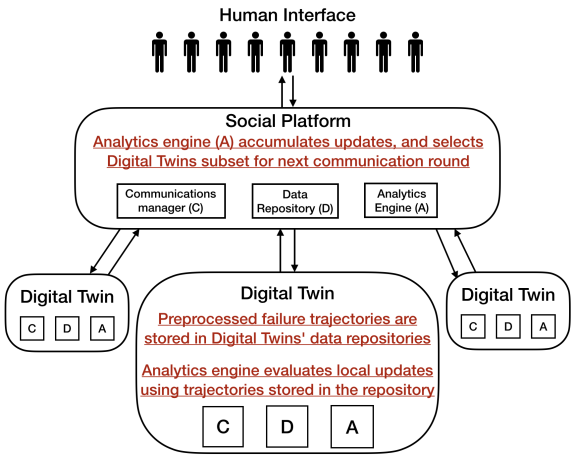

Fig. 3: This figures presents a schematic description of computations and data stored at architectural levels. Only three digital twins are shown for ease of presentation, but the actual number of digital twins equals number of assets comprising the cluster.

Similarly, tasks performed by the analytics engine of the social platform are to aggregate updates from the participating digital twins, and to select the subset of digital twins for the following communication round. A schematic diagram indicating the computations (red underlined text) involved at corresponding levels of the architecture is shown in Figure 3. Further implementation details about the experiments are explained in Section 4.

3.3 Developmental Specification

Python 3.6 and its standard libraries including Pandas and Numpy were used to preprocess the dataset. While TensorFlow Federated framework could be directly used to implement FedAvg, owing to the nature of its beta release, TensorFlow Federated framework is extremely slow and does not enable necessary modifications of the hyper parameters involved. Therefore, Python's Socket library was used to develop a network of digital twins and the social platform. To enable parallel computation, digital twins were run on separate processor threads using Python's Multithreading library. Keras library with Tensorflow backend was used to develop and train RNNs, and NVIDIA Tesla P100 server processor with 3584 CUDA cores GPU was used to perform the experiments discussed here.

4 Experiments

This section explains the various experimental cases that were performed to analyse FedAvg for collaborative prognosis. Experiment cases were designed to study the effect of various FedAvg parameters on the training process, and also to serve as an example application.

4.1 Experiment Cases

A Recurrent Neural Network (RNN), specifically consisting of one Long-Short Term Memory (LSTM) layer, was used in the experiments as prediction model. RNN is a special type of ANNs, where the outputs of certain neurons are included with their inputs. This feature enables the RNNs to understand the time dependency of trending features, and therefore make them well suited for time series prediction applications like prognosis.

The RNN used for experiments here comprised of three intermediate layers, containing 12*25*10 neurons respectively, where the layer containing 12 neurons was the LSTM layer, and the rest were standard feed forward neurons. The $tanh$ activation functions were used at every neuron. Owing to the $tanh$ activation function, the range of the output neuron was constrained between 0 to 1. Therefore, the same MinMax scaler used to downscale the training data was used to upscale the output of the RNN.

Table 2 Values of hyper parameters across various experiment cases.

Hyper parameter	Value
C	$\{0.1, 0.25, 0.5, 1\}$
E	$\{1, 5, 10, 20\}$
λ	$\{0.01, 0.05, 0.1\}$
n	$\{10, 35, 70, 138\}$

Table 3 Total assets and number of failures at individual assets corresponding to various number of total failures

Total Failures (n)	**Total Assets** (m)	**Failures per individual Assets** (n_k)
10	10	[1,1,1,1,1,1,1,1,1,1]
35	16	[1,1,2,3,3,1,2,3,3,2,1,3,3,1,3]
70	17	[4,7,4,7,5,5,5,5,7,3,3,7,1,3,3,1,2]
138	15	[6,12,3,13,7,12,13,6,9,5,13,8,6]

Effects of the hyper parameters including the participating fraction of assets (C), epochs per asset participating in the update step (E), learning rate (λ) of the RNN during local updates, total failures in the fleet (n), and the optimiser of the RNN were studied by varying them across different values. Moreover, decaying learning rate with subsequent communication rounds was also experimented and compared with a constant learning rate. Table 2 summarises the values of the above parameters studied across the experiment cases. Global and local objective functions were both aiming to minimise the mean absolute difference between real and predicted RUL values for trajectories in training dataset.

For each set of hyper parameter values, the RNN was trained for 600 communication rounds, and its mean absolute error of predictions for test failure trajectories was evaluated at the end of each round.

4.2 Fleet Simulation

From the remaining 138 trajectories obtained after preprocessing explained in Section 3.1, the initial n trajectories were used for the corresponding experiment cases described in Section 4.1. However, to replicate failures distributed across multiple assets, these n trajectories were further segmented into smaller groups of trajectories and stored in data repositories of separate digital twins.

Let these digital twins be indexed using $k \in 1, 2, ...,$ therefore $|k| = m$, where m is the number of assets in the cluster. Each of the digital twins holds n_k trajectories, where the integer n_k is randomly selected as $n_k \in [1, \text{int}(n/10)]$. Concretely, this resembles a cluster where m assets have failed due to a given failure mode, with varying number of failure occurrences. The goal was to train the RNN using this dataset of total n failures, distributed across m assets. The assets (m) and number of failures of individual assets (n_k) corresponding to different values of total failures (n) are presented in Table 3.

All possible permutations of parameter values listed in Table 2 were analysed for FedAvg training. Analysis included recording the global model's mean of the absolute prediction error for the test data after every communication round. Rate and extent error reduction in the end model performance were studied for all sets of parameter values.

5 Discussion

This Section discusses the effect of parameters deduced from the experimental results, and presents the corresponding performance plots explaining those effects. While plots for only certain parameter values are included for ease of presentation, the corresponding

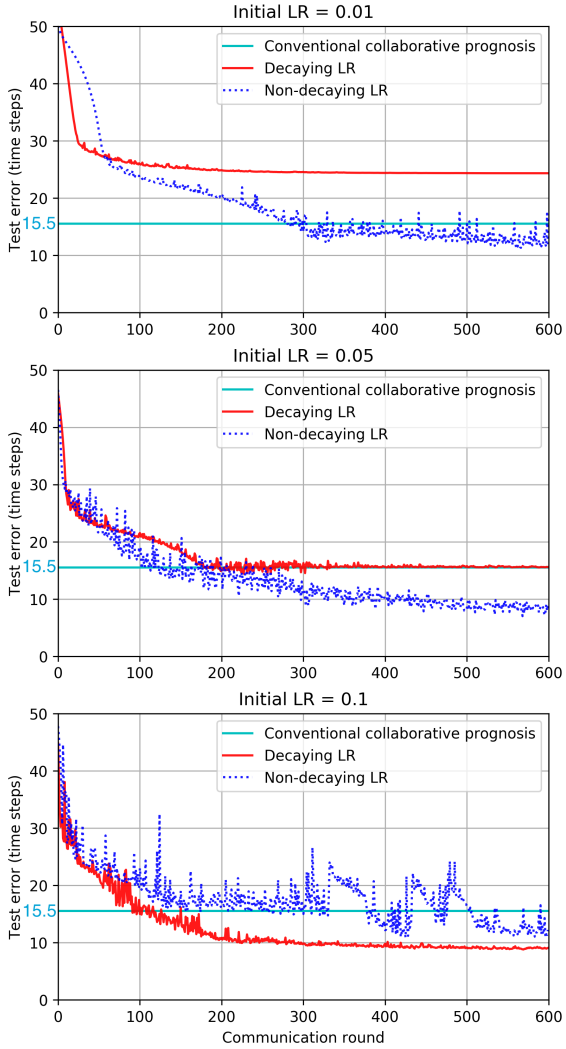

Fig. 4: Effect of learning rate on model performance. Parameters $(C, E, n) = (0.5, 5, 70)$ with SGD optimiser. LR = Learning Rate.

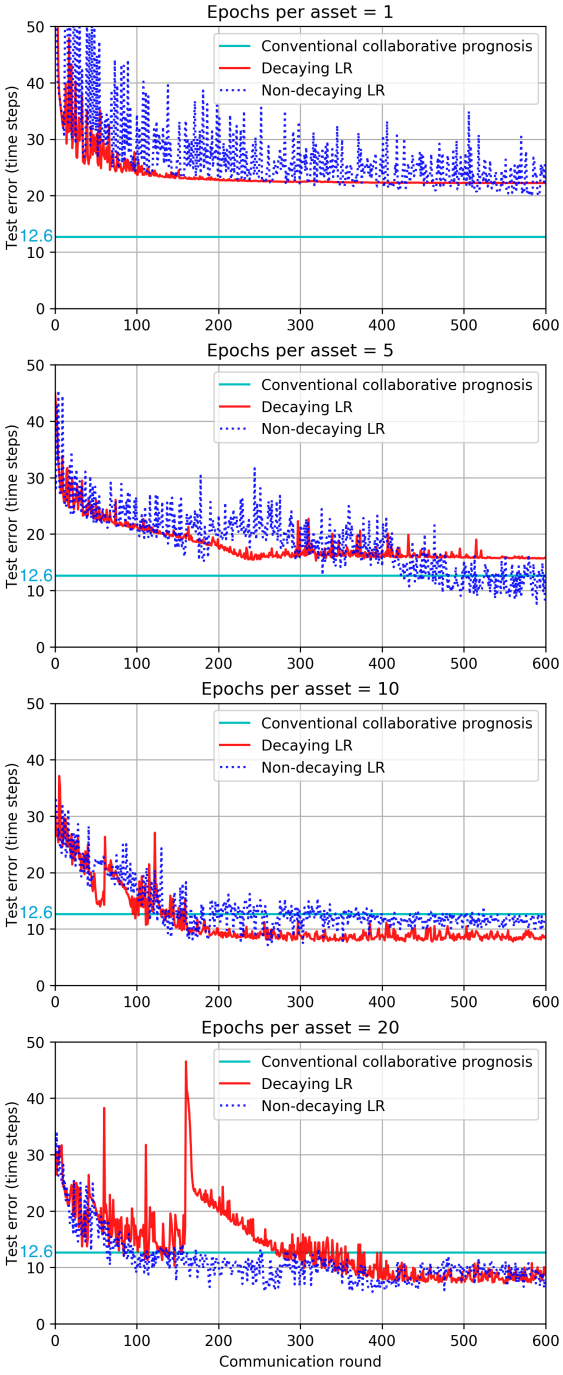

Fig. 5: Effect of epochs per asset on model performance. Parameters $(C, \lambda, n) = (0.25, 0.1, 138)$ with SGD optimiser. LR = Learning Rate.

conclusions hold true across all permutations. Parameter values associated with the plots in following subsections are all mentioned in their figure captions.

Corresponding model performances achieved with conventional collaborative prognosis, which involved sharing failure data across assets, for same values of total failures n and the optimisers are also shown on those plots. RNN with same architecture as the one used for FedAvg was used for conventional collaborative prognosis. It was trained using an optimal set of hyper parameters, and was trained until its error for test data did not decrease any further. The same test data, and mean absolute error of predictions, were used to evaluate the performance of model trained using conventional collaborative prognosis as well. The horizontal cyan lines in Figures 4 to 8 indicate test errors while using conventional collaborative prognosis for training the same prediction models and with same failure trajectories as FedAvg.

5.1 Effect of Decaying Learning Rate (λ)

Shown in Figure 4 are performances of models trained using a decaying learning rate, alongside the same models trained using a constant learning rate. It is observed that allowing λ to decay after every communication round stabilises the model's test error while training. A constant decay of 0.99 was implemented during the experiments. Model training was found to be comparatively stable for decaying learning rates than constant learning rate across all parameter combinations, similar to that shown in Figure 4.

However, if the initial λ is not sufficiently high enough, end model performances tend to a higher test error than the one trained using non-decaying learning rate. This is observed in the performance

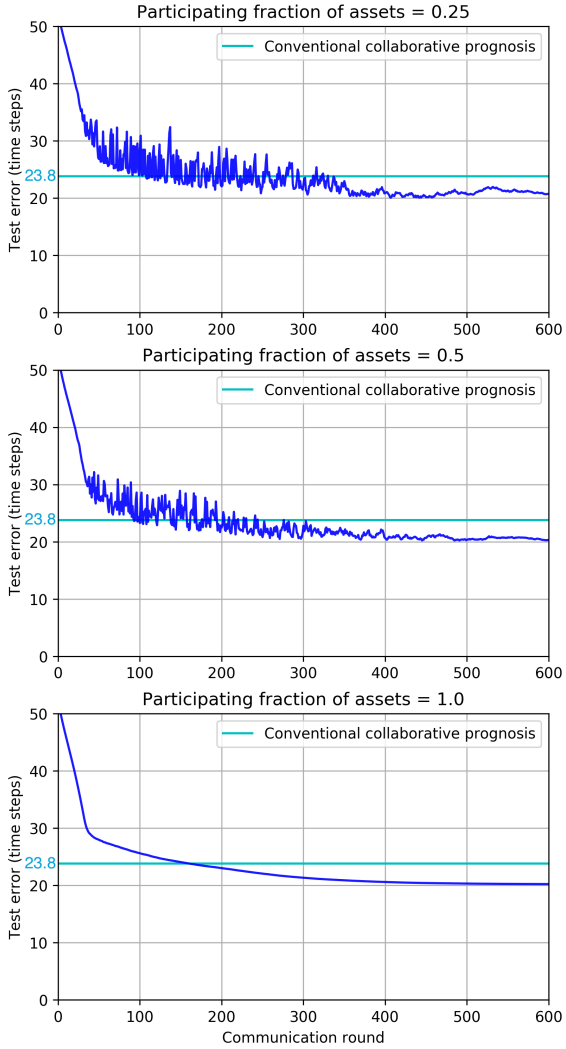

Fig. 6: Effect of participating fraction of assets on model performance. Parameters $(n, \lambda, E) = (10, 0.01, 5)$ with SGD optimiser.

plots shown in Figure 4, where for lower initial λ, models converge to substandard performance compared to non-decaying learning rate.

Plots for non-decaying λ are shown using blue dotted line and for decaying λ using red solid line in Figure 4. Their initial learning rates are also mentioned on the same plots, and the corresponding parameter values, which were constant for all three plots, are mentioned in the caption.

5.2 *Effect of Epochs per Asset* (E)

It can be observed from the plots presented in Figure 5, that the test errors during training process decrease and becomes more stable as E increases. The test error was found to stabilise with increasing E value, with other parameters kept constant. Moreover, as the number of epochs per asset are increased, models converge to much lower test errors, compared to the same ones stabilised using learning rate decay. In Figure 5, constant learning rate while training is shown using blue dotted line, and decaying rate with red solid line.

The marginal improvement in end model performance, as it was expected and also observed in Figure 5, however decreases with increasing E. Therefore, the optimal E value corresponds to its minimum value which stabilises model training and results in an acceptable end model performance.

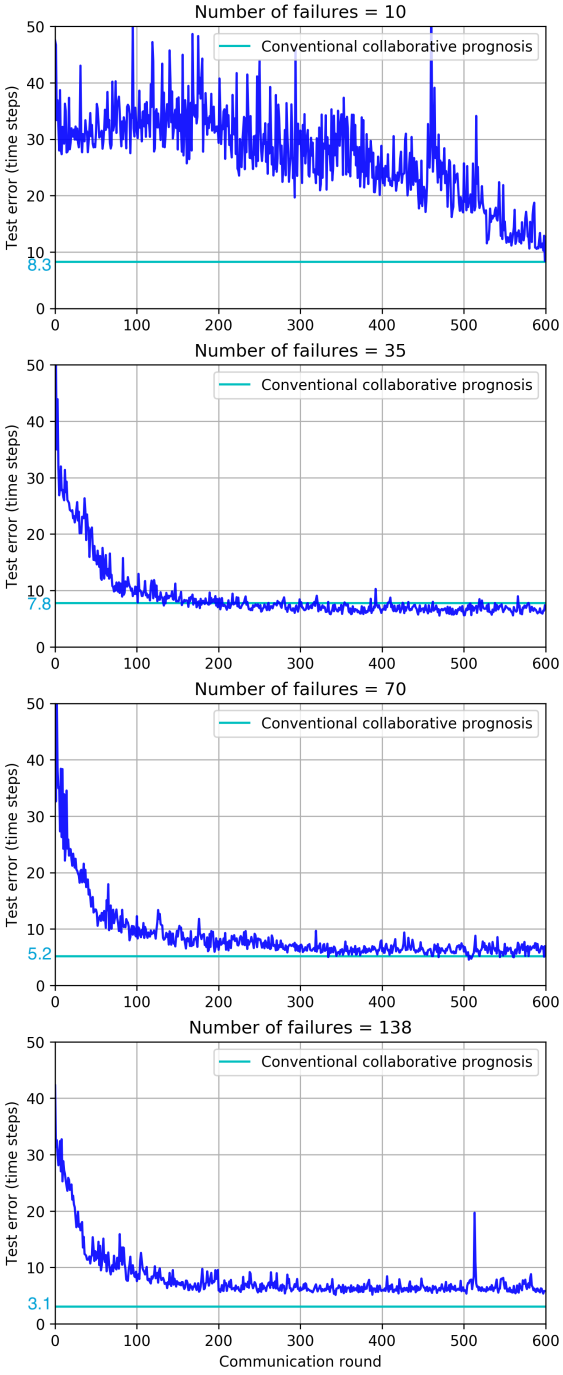

Fig. 7: Effect of total failures on model performance. Parameters $(C, \lambda, E) = (0.25, 0.1, 5)$ with Adam optimiser.

5.3 *Effect of Participating Fraction of Assets* (C)

Experiments showed that increasing C had no substantial effect on the end model performances. However, test errors stabilised with increasing C value. This is presented in Figure 6.

IET Research Journals, pp. 1–10

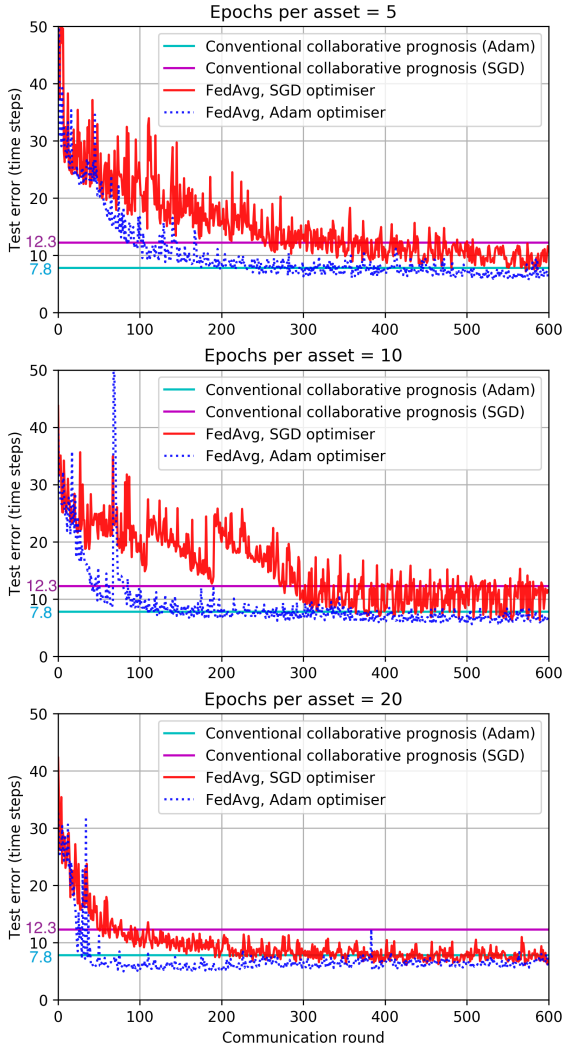

Fig. 8: Effect of the optimiser on model performance. Parameters $(C, \lambda, n) = (0.25, 0.1, 35)$. Performances of models trained using conventional collaborative prognosis, having Adam and SGD optimisers are marked with cyan and magenta horizontal lines respectively.

5.4 *Effect of Total Failures* (n)

The experiments showed that, similar to conventional collaborative prognosis (see [9], pp. 600-601), a minimum number of failure trajectories were necessary for prediction models to achieve acceptable accuracies. The test error for $n = 10$ is comparatively erratic and higher than other values of n, which decreases with increasing n. But unlike conventional prognosis where model performance continuously increases with increasing number of failure trajectories, FedAvg's end model performance saturates after a certain value of n, which for experiments discussed here is 35. Figure 7 illustrates the effect of n on model training.

5.5 *Effect of the Optimiser*

Two popular optimisers- Adam and SGD- were implemented for the experiments discussed here. SGD is a classic optimiser for ANNs, and pioneering applications of FedAvg involved using SGD. Adam is a comparatively newer optimiser than SGD, which is in fact a modification of SGD that uses adaptive learning rate based on training data and error [37].

A comparison between Adam and SGD optimisers for FedAvg is shown in Figure 8 for various E values, where SGD optimiser is represented by the red solid line, and Adam optimiser by blue dotted line. It was found in the experiments that Adam performed better than SGD in most experiment cases. However, for either optimisers, models trained using FedAvg could attain their corresponding performances attained by conventional collaborative prognosis. This confirms that FedAvg is applicable for collaborative prognosis.

6 Conclusions

This paper highlights those challenges which hinder implementation of distributed data driven prognosis techniques, specifically collaborative prognosis. Of the general challenges faced by distributed optimisation problems, conventional collaborative prognosis is incapable of addressing problems of data security and communication efficiency. It is identified that specifically the failure data exchange step of collaborative prognosis hinders its application for real world industries.

Also proposed in this paper is the application of Federated Averaging to replace the failure data exchange part of collaborative prognosis. Authors have demonstrated that FedAvg is applicable for asset prognosis, and that it is capable of realising collaborative prognosis. This follows from the observation in figures 4 to 8, that performances of models trained using FedAvg have improved during the training process using FedAvg, and in many cases in fact surpassed those of models trained using conventional collaborative prognosis. By avoiding failure data exchange within assets, FedAvg also has added benefits of securing asset data and of reducing network communication costs. Addressing the problem of localised training without the need for failure data to leave assets is also deemed beneficial for data driven prognosis techniques in general, that rely on failure trajectories distributed across several assets.

As shown in figures 4 to 8, values of parameters including (C, E, λ), just like any other machine learning technique, must however be carefully tuned to achieve optimal model performance. While Section 5 discusses the effect each parameter is expected to have on FedAvg training, optimal permutation of parameters is dependent on the application and best deduced by its corresponding operators.

While the challenges faced by the conventional collaborative prognosis techniques were mentioned in [38], this paper discusses them in more detail, proposes FedAvg as a solution to address those challenges, and presents the results from extensive experiments performed to analyse model performances for the proposed application.

7 Future Research Directions

The proposed application and experiments also make way for exciting possibilities related to both general FL, and asset prognosis research:

1. As discussed in Section 2.3, FedAvg is applicable for IID data only. While IID failure trajectories are identified by the clustering step, it does not make the collaborative prognosis pipeline truly localised. This is because the server performs clustering by analysing failure trajectories across all assets. Therefore, asset data must be shared with the server. Future research can improve upon the current FedAvg implementation to make it capable of training prediction models for IID data, without the need for data to leave assets at all. Some inspiration for this can arise from FL literature like [23, 29].

2. Another challenge is to implement the trained models in real time. When an asset is found to deviate from its normal behaviour, it is seldom possible to identify the impending failure and its prediction model. Collaborative prognosis requires an asset to operate in a

given condition for certain period before it can be clustered with similar assets. This makes it incapable of predicting failures of a newly introduced asset. Future research can involve automated real time identification of most suitable prediction model for failing assets. Neural network ensembles, where their prediction is also associated with confidence, can possibly address this challenge [39, 40].

3. It is theoretically shown in Section 2.4 that federated learning reduces the data transferred within the asset network, and therefore reduces the communication costs to the operator. Future work can investigate the extent of this reduction for various industrial applications, to identify important tradeoffs between model performance and overall data transfer.

4. Lastly, an important follow-up task is to implement FedAvg for real world asset data. Maximum 148 IID failure trajectories could be obtained from the C-MAPSS dataset. This is because C-MAPSS dataset has only 148 instances where assets operating in similar operating conditions incur same failure type. Real world data could comprise of higher number of failure trajectories for more failure types. This could enable analysing FedAvg performance for different failure types.

8 Acknowledgements

This research was funded by the EPSRC and BT Prosperity Partnership project: Next Generation Converged Digital Infrastructure, grant number EP/R004935/1. This research was also funded by Siemens Industrial Turbomachinery, Lincoln, UK LN5 7FD.

9 References

1 D. Goyal and B. S. Pabla, "Condition based maintenance of machine tools-A review," *CIRP Journal of Manufacturing Science and Technology*, vol. 10, pp. 24–35, aug 2015.
2 A. Neely, "Exploring the financial consequences of the servitization of manufacturing," *Operations Management Research*, vol. 1, pp. 103–118, dec 2008.
3 M. Schwabacher and K. Goebel, "A Survey of Artificial Intelligence for Prognostics," in *AAAI Fall Symposium: Artificial Intelligence for Prognostics*, 2007.
4 A. K. Jain and B. K. Lad, "Data driven models for prognostics of high speed milling cutters," *International Journal of Performability Engineering*, 2016.
5 J. Lee, C. Jin, Z. Liu, and H. D. Ardakani, "Introduction to data-driven methodologies for prognostics and health management," in *Probabilistic Prognostics and Health Management of Energy Systems*, pp. 9–32, Springer, 2017.
6 A. K. Jain and B. K. Lad, "Dynamic optimization of process quality control and maintenance planning," *IEEE Transactions on Reliability*, vol. 66, pp. 502–517, jun 2017.
7 A. Salvador Palau, Z. Liang, D. Lütgehetmann, and A. K. Parlikad, "Collaborative prognostics in Social Asset Networks," *Future Generation Computer Systems*, vol. 92, pp. 987–995, mar 2019.
8 K. Bakliwal, M. H. Dhada, A. S. Palau, A. K. Parlikad, and B. K. Lad, "A Multi Agent System architecture to implement Collaborative Learning for social industrial assets," *IFAC-PapersOnLine*, vol. 51, pp. 1237–1242, jan 2018.
9 A. Salvador Palau, M. H. Dhada, K. Bakliwal, and A. K. Parlikad, "An Industrial Multi Agent System for real-time distributed collaborative prognostics," *Engineering Applications of Artificial Intelligence*, vol. 85, pp. 590–606, oct 2019.
10 A. S. Palau, K. Bakliwal, M. H. Dhada, T. Pearce, and A. K. Parlikad, "Recurrent Neural Networks for real-time distributed collaborative prognostics," in *2018 IEEE International Conference on Prognostics and Health Management, ICPHM 2018*, Institute of Electrical and Electronics Engineers Inc., aug 2018.
11 A. Salvador Palau, M. H. Dhada, and A. K. Parlikad, "Multi-agent system architectures for collaborative prognostics," *Journal of Intelligent Manufacturing*, vol. 30, pp. 2999–3013, dec 2019.
12 T. Wang, J. Yu, D. Siegel, and J. Lee, "A similarity-based prognostics approach for remaining useful life estimation of engineered systems," in *2008 International Conference on Prognostics and Health Management, PHM 2008*, 2008.
13 Y. Lin, S. Liu, and S. Huang, "Selective sensing of a heterogeneous population of units with dynamic health conditions," *IISE Transactions*, vol. 50, pp. 1076–1088, dec 2018.
14 O. F. Eker, F. Camci, and I. K. Jennions, "A Similarity-based Prognostics Approach for Remaining Useful Life Prediction," in *Second European Conference of the Prognostics and Health Management Society*, PHM Society, may 2014.
15 D. Mcfarlane, "Industrial Internet of Things Applying IoT in the Industrial Context," tech. rep., Institute for Manufacturing, University of Cambridge, 2019.
16 P. Leitão and S. Karnouskos, *Industrial Agents: Emerging Applications of Software Agents in Industry*. Elsevier Inc., mar 2015.
17 B. Saha, S. Saha, and K. Goebel, "A Distributed Prognostic Health Management Architecture," in *Sixth International Conference on Condition Monitoring and Machinery Failure Prevention Technologies – CM/MFPT*, 2009.
18 A. Gilchrist and A. Gilchrist, "IIoT Reference Architecture," in *Industry 4.0*, pp. 65–86, Apress, 2016.
19 A. R. Sadeghi, C. Wachsmann, and M. Waidner, "Security and privacy challenges in industrial Internet of Things," in *Proceedings - Design Automation Conference*, vol. 2015-July, Institute of Electrical and Electronics Engineers Inc., jul 2015.
20 C. E. Siemieniuch and M. A. Sinclair, "On complexity, process ownership and organisational learning in manufacturing organisations, from an ergonomics perspective," *Applied Ergonomics*, vol. 33, pp. 449–462, sep 2002.
21 J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated Learning: Strategies for Improving Communication Efficiency," *arXiv preprint arXiv:1610.05492*, oct 2016.
22 Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, pp. 1–19, jan 2019.
23 T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated Learning: Challenges, Methods, and Future Directions," *arXiv preprint arXiv:1908.07873*, aug 2019.
24 G. D. Ranasinghe, T. Lindgren, M. Girolami, and A. K. Parlikad, "A Methodology for Prognostics under the Conditions of Limited Failure Data Availability," *IEEE Access*, vol. 7, pp. 183996–184007, 2019.
25 R. Bekkerman, M. Bilenko, and J. Langford, *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press, 2011.
26 K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. B. McMahan, T. Van Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards Federated Learning at Scale: System Design," *arXiv preprint arXiv:1902.01046*, feb 2019.
27 L. Huang, Y. Yin, Z. Fu, S. Zhang, H. Deng, and D. Liu, "LoAdaBoost:Loss-Based AdaBoost Federated Machine Learning on medical Data," *arXiv preprint arXiv:1811.12629*, nov 2018.
28 A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated Learning for Mobile Keyboard Prediction," *arXiv preprint arXiv:1811.03604*, nov 2018.
29 T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated Optimization in Heterogeneous Networks," *arXiv preprint arXiv:1812.06127*, dec 2018.
30 Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, may 2015.
31 H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, feb 2016.
32 A. Saxena and K. Goebel, "Turbofan Engine Degradation Simulation Data Set," tech. rep., NASA Ames Research Center, Moffett Field, CA, 2008.
33 A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *2008 International Conference on Prognostics and Health Management, PHM 2008*, 2008.
34 M. Herrera, M. Pérez-Hernández, A. Kumar Parlikad, and J. Izquierdo, "Multi-Agent Systems and Complex Networks: Review and Applications in Systems Engineering," *Processes*, vol. 8, p. 312, mar 2020.
35 B. X. Yong and A. Brintrup, "Multi agent system for machine learning under uncertainty in cyber physical manufacturing system," in *International Workshop on Service Orientation in Holonic and Multi-Agent Manufacturing*, pp. 244–257, Springer, 2019.
36 M. Wooldridge, *An introduction to multiagent systems*. John Wiley & Sons, Ltd, 2009.
37 D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, International Conference on Learning Representations, ICLR, dec 2015.
38 M. H. Dhada, A. S. Palau, and A. K. Parlikad, "Federated Learning for Collaborative Prognosis," in *International Conference on Precision, Meso, Micro, and Nano Engineering*, (IIT Indore, India), 2019.
39 T. Pearce, M. Zaki, A. Brintrup, and A. Neely, "High-Quality Prediction Intervals for Deep Learning: A Distribution-Free, Ensembled Approach," in *35th International Conference on Machine Learning, ICML 2018*, vol. 9, pp. 6473–6482, International Machine Learning Society (IMLS), feb 2018.
40 J. K. Ambrosio, B. M. Brentan, M. Herrera, E. Luvizotto, L. Ribeiro, and J. Izquierdo, "Committee machines for hourly water demand forecasting in water supply systems," *Mathematical Problems in Engineering*, vol. 2019, 2019.