Accepted version, published version available at 10.1109/ETFA46521.2020.9211903

Continual Learning of Fault Prediction for Turbofan Engines using Deep Learning with Elastic Weight Consolidation

Benjamin Maschler[1]
Institute of Industrial Automation and Software Engineering
University of Stuttgart
Stuttgart, Germany
benjamin.maschler@ias.uni-stuttgart.de

Hannes Vietz[1]
Institute of Industrial Automation and Software Engineering
University of Stuttgart
Stuttgart, Germany
hannes.vietz@ias.uni-stuttgart.de

Nasser Jazdi
Institute of Industrial Automation and Software Engineering
University of Stuttgart
Stuttgart, Germany
nasser.jazdi@ias.uni-stuttgart.de

Michael Weyrich
Institute of Industrial Automation and Software Engineering
University of Stuttgart
Stuttgart, Germany
michael.weyrich@ias.uni-stuttgart.de

***Abstract*— Fault prediction based upon deep learning algorithms has great potential in industrial automation: By automatically adapting to different usage contexts, it would greatly expand the usefulness of current predictive maintenance solutions. However, restrictions regarding the centralized accumulation of data necessary for such automatic adaption call for a distributed approach to training these algorithms. Therefore, in this paper, a continual learning based algorithm for fault prediction is presented, allowing for distributed, cooperative learning by elastic weight consolidation. This algorithm is then evaluated on a large NASA turbofan engine dataset and shows promising results regarding the performant training on decentral sub-datasets for industrial automation scenarios.**

Keywords—Continual learning, Elastic weight consolidation, Fault prognostics, Industrial Automation, Machine learning, Neural networks, Prediction methods, Transfer learning

I. Introduction

The trend of industrial digitalization has led to an increasing system complexity, which in turn brought about an increased frequency and a reduced foreseeability of ever more complex faults. However, the same developments combined with the concepts of Industry 4.0 or the Industrial Internet of Things resulted in a widespread availability of high quality, high-resolution data [1]. This data is the basis for fault diagnosis and prognostics required for predictive maintenance – a possible solution to the aforementioned problem. Yet, many of these approaches still require a great level of manual adaption to the respective scenarios [2–4] – a deed oftentimes impossible to carry out, because the specific operating conditions constituting those scenarios might be unknown to the provider of predictive maintenance solutions.

Deep learning based fault prediction offers automatic adaptability to those different usage scenario, e.g. of the same machine at different operating sites, to allow a more flexible utilization of such approaches [2, 5, 6]. This, however, requires the accumulation of large amounts of training data describing the different scenarios [7–9], which the customers of deep learning based predictive maintenance solutions, usually the owners of the machines, might not be willing to share due to privacy or industrial espionage concerns, technical or legal reasons.

Problem statement: Today, deep learning based fault prediction requires large amounts of data in a central data lake to facilitate training. In many scenarios, this is not possible due to legal or technical reasons which prevent such merging of datasets. Therefore, an approach for the disjoint processing of deep learning based fault prediction is needed.

Continual learning algorithms could solve this problem, because they facilitate performant training on small, decentral datasets [9–12]. One such algorithm is the so-called *Elastic Weight Consolidation* (EWC), which enhances a conventional deep learning architecture already suiting the respective industrial use case for a centralized dataset [10].

Structure: This paper presents an industrial application case study in *section II A*, discusses the state-of-the-art in deep learning based fault prediction in *section II B* and gives an overview of continual learning and EWC-based knowledge transfer in *section II C*. Then, a conventional deep learning architecture for fault prediction based upon these findings is developed in *section III A*. In *section III B*, it is expanded to incorporate EWC to enable effective learning on smaller, decentral datasets without the need for centralized (cloud) storage. Finally, both algorithms are evaluated in *section IV* and a conclusion is drawn in *section V*.

II. Related Work

This section introduces related work regarding a suitable dataset, deep learning based turbofan engine degradation prognostics as well as continual learning and EWC. It closes with a brief conclusion. Regarding the degradation prognostics, comparison of recently published approaches is presented. For

[1] These authors contributed equally to this publication.

EWC, a general description as well as recent implementation examples are given.

A. *Turbofan Degradation Dataset*

A well-used dataset for fault prediction using deep learning is the *Turbofan Engine Degradation Simulation Data Set* (TEDSDS) by NASA [13]. It is based on the Commercial Modular Aero-Propulsion System Simulation developed by [14] and includes four simulated datasets (FD1 to FD4). Each dataset consists of several dozens of individual engines' time series. Each individual engine's data starts with a varying degree of initial wear and manufacturing variation. The engines are then run to failure in the training data. For the test data, the time series stop before a failure occurs, but instead the true *Remaining Useful Life* (RUL) is given. Every data point consists of a unit number, a timestamp, three operational parameters and 21 different sensor measurements of which some are constant. The different datasets vary in complexity as the number of operating conditions changes between one (FD1 and FD3) and six (FD2 und FD4).

As performance metrics, [15] proposes the *Root Mean Square Error* (RMSE) and a *Prognostic Health Management Error* (PHME) function (1), where A_t is the actual RUL, P_t is the predicted RUL, n is the number of predicted RULs, $a_1 = 13$ and $a_2 = 10$. Because the safety-critical nature of turbofans is better represented by the PHME punishing overestimations of the RUL heavier than underestimations, it is considered the more important metric (see Fig. 1).

$$\text{PHME} = \begin{cases} \sum_1^n e^{-\frac{A_t - P_t}{a_1}} - 1 \text{ for } (A_t - P_t) < 0 \\ \sum_1^n e^{\frac{A_t - P_t}{a_2}} - 1 \text{ for } (A_t - P_t) \geq 0 \end{cases} \quad (1)$$

B. *Deep Learning Architectures for Turbofan Degradation Prognostics*

Many variants of deep learning algorithms have been applied to the TEDSDS in recent years (see Table 1). Studies discussed here utilize *Convolutional Neural Networks* (CNN), *Long-Short Term Memory networks* (LSTM) or *Deep Belief Networks*. A piecewise linear RUL labeling was used, which acknowledges that in the early stage of a turbine's runtime no degradation is discernible. This approach is used by all publications discussed here.

Reference [16] applied a CNN to RUL prediction. They used the sensor data of the turbofans and one handcrafted feature

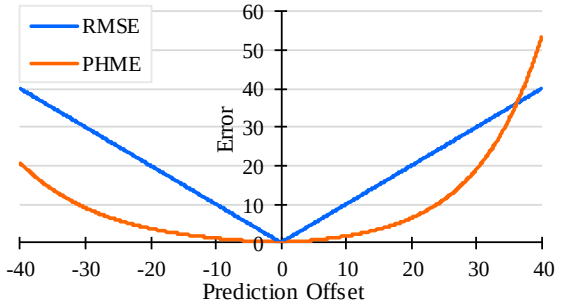

Fig. 1. Simple comparison of RMSE and PHME in case of underestimation (left) and overestimation (right) of RUL

TABLE I. RECENT DEEP LEARNING LITERATURE RESULTS: PHME (UPPER VALUE) AND RMSE (LOWER VALUE) ON THE TEDSDS

Algorithm	Dataset			
	FD1	***FD2***	***FD3***	***FD4***
CNN [16]	1286 18.45	13570 30.30	1596 19.82	7886 29.16
CNN [17]	274 12.61	10412 22.36	284 12.64	12466 23.31
Ensemble Deep Believe Network [18]	334 15.04	5585 25.05	422 12.51	6558 28.66
LSTM [21]	338 16.14	4450 24.49	852 16.18	5550 28.17
Restricted Boltzmann Machine + LSTM [22]	231 12.56	3366 22.73	251 12.10	2840 22.66
Autoencoder + LSTM [23]	261 13.63	- -	- -	- -

along a 15-cycle wide timed window as input for their two-dimensional convolution layers. Two-dimensional convolutions allow the CNN to learn correlations along the feature dimension as well as the time dimension. The architecture is a conventional CNN with average pooling and an added fully connected layer that predicts the RUL. Reference [16]'s approach showed improvements compared to *Support Vector Machines*, *Relevance Vector Machines* and *Multilayer Perceptrons*.

Reference [17] utilized an approach similar to [16]. However, they added no features by hand, removed the pooling layers and reduced the convolution dimensions to a single one, which was applied only in the time dimension. Through these changes and a deeper CNN they were able to achieve better results than [16].

Reference [18] used an ensemble of *Deep Believe Networks* consisting of stacked *Restricted Boltzmann Machines*, which are each unsupervisedly trained with contrastive divergence. The *Deep Believe Networks* are then supervisedly trained with backpropagation. Ensembles are a combination of several (base)-models forming one better performing model. Previous work on ensembles showed improvements in generalization [19] and mitigating catastrophic forgetting [20]. Reference [18] used a *Multiple Objective Evolutional Algorithm Based on Decomposition* to train different *Deep Believe Networks*. More specifically, they use the evolutional algorithm to evolve the hyper- and training parameters of the *Deep Believe Networks*. After a certain number of generations, several evolved *Deep Believe Networks* then form the ensemble. A *Multiple Objective Evolutional Algorithm Based on Decomposition* can have multiple objective functions instead of just one like conventional evolutional algorithms. Reference [18] used this to evolve not only for accuracy but also for diversity of the individual *Deep Believe Networks*. Their results show impressive prediction qualities compared with other *Deep Believe Networks* and that they can outperform learning algorithms like *Multilayer Perceptrons* and *Support Vector Machines*.

Reference [21] chose a 2-layer deep LSTM, which are widely used for sequential data, followed by two fully connected layers to predict the RUL. Their results are promising and consistent over the datasets, outperforming CNNs, *Multilayer Perceptrons* and *Support Vector Machines*.

The architecture proposed by [22] is similar, but they added another fully connected layer before the LSTMs. This layer was unsupervisedly pretrained with a *Restricted Boltzmann Machine*, like one layer of a *Deep Believe Network*. Additionally, they performed a hyperparameter optimization with a genetic algorithm. This semi-supervised combination lead to very good results, beating all other deep learning algorithms discussed here.

Instead of a semi-supervised training, [23] placed an *Autoencoder* before the LSTM layers. *Autoencoders* are well-established, unsupervisedly trained feature extraction algorithms. Reference [23] used bidirectional LSTM layers, whose predictions can be influenced by later time steps. Bidirectionality is often used in natural language processing, so later parts of a sentence can change predictions of earlier parts, however, this does not seem useful for fault prediction. Their published result on one turbofan engine looks promising, but it remains unclear whether their approach translates well to the other datasets.

However, all the approaches discussed require large datasets to train on. In order to allow for the effective training on smaller, decentral datasets without causing catastrophic forgetting, these approaches need to be combined with continual learning methods.

C. Continual Learning and Elastic Weight Consolidation

The term 'continual learning' describes a set of approaches aimed at transferring knowledge from one or more source tasks to a target task in order to train a deep learning algorithm capable of solving source and target tasks. This is sometimes called 'multi-tasking' or 'incremental learning' [24].

Continual learning approaches can be divided into three approach categories: architectural, rehearsal and regularization approaches [24]. For a practical application in solving the problem of learning on not to be merged, but somewhat similar datasets, one approach category is more promising than the other two: While rehearsal approaches bear the risk of disclosing too much information taken directly from the datasets themselves and architectural approaches strive on more loosely related tasks, regularization approaches using altered loss functions and rather abstract meta-data in order to solve more closely related tasks appear best suited for the problem at hand. Among the older, more widely discussed and applied regularization approaches is EWC, which is therefore chosen for the approach proposed in this article.

Reference [10] introduced EWC as a method to overcome the problem of a neural network's loss of knowledge concerning previously learned tasks while training new tasks in multi-tasking problems, the so-called catastrophic forgetting [25]. It is based on the fact that much more than one set of weights and biases θ characterizing a trained deep neural network will result in the same performance of said network [26]. This allows for a possible solution θ_B^* of a task B to be close to a previously found possible solution θ_A^* of a task A and therefore enables a neural network to incorporate both sets of weights and biases in a manner that solves both tasks. This "closeness" of the possible solutions is achieved by constraining the change of parameters most important to the performance in the previously learned task A by adding a quadratic penalty to the loss function (2): $L_B(\theta_{AB})$ is the loss for task B without that penalty, λ defines the importance of the old task compared to the new one, F is the diagonal of the Fisher information matrix and i labels each individual parameter.

$$L(\theta_{AB}) = L_B(\theta_{AB}) + \sum_i \lambda_A \cdot F_{A,i}\left(\theta_{AB,i} - \theta^*_{A,i}\right)^2 \qquad (2)$$

Correspondingly, the loss function for a third task C would be (3).

$$L(\theta_{ABC}) = L_C(\theta_{ABC}) + \sum_i [\lambda_A \cdot F_{A,i}\left(\theta_{ABC,i} - \theta^*_{A,i}\right)^2 + \lambda_B \cdot F_{B,i}\left(\theta_{ABC,i} - \theta^*_{AB,i}\right)^2] \qquad (3)$$

In a reply to [10], [27] argues that starting with the third task, any intermediate parameter estimates are unnecessary as the information contained within them is already represented by the latest parameter estimate. However, as pointed out by [28] and verified by experiments carried out by [29], these intermediate parameter estimates improve the "remembrance" of older tasks and thereby the algorithms performance.

Since its initial presentation, EWC has been put on trial in different domains and scenarios:

In the field of medical radiology, [30] examined the performance of an EWC-based algorithm on different image segmentation tasks. The images were MRI scans of human brains on which the algorithm outdid all other, non-continual approaches except for one where all tasks were trained simultaneously. However, [30] saw "significant space for further research" as the performance was still significantly worse than that of single-task ML. Reference [31], too, examined a segmentation scenario. They found that EWC "restricted the degree of catastrophic forgetting but also the ability to adopt to the new domain." Similarly, they saw a need for "more extensive evaluation."

In the field of machine translation, [32] examined the performance of an EWC-based algorithm regarding machine translation of sentences from news and patents. The data was from different, domain-typical databases, e.g. multilingual patent databases. EWC outperformed other, state-of-the-art approaches in five out of six sub-tasks. Therefore, [32] saw it fit to be "used in practical situations."

To the authors' knowledge, no EWC trial has been published for the field of RUL prediction or predictive maintenance.

D. Conclusion of Related Work

The benchmark dataset used in this paper is the TEDSDS by NASA. Its pre-defined test data ensures a comparability of results with the many publications in which it is used as well. As deep learning method, LSTM is chosen due to the high prediction performance, as discovered by earlier publications, and its superior adaptability to different data formats compared to other approaches. To allow for the performant training on decentral, smaller datasets, EWC that has shown promising results in image segmentation and machine translation is introduced. Not having been applied to the field of predictive

maintenance yet, it is the aim of this paper to investigate the potential of EWC in fault prediction.

III. Methodology

In this section, the proposed deep learning architectures for RUL prediction are presented. First, the architecture used without EWC and the associated hyperparameter search are discussed. Next, the proposed architecture used with EWC is described.

A. *Proposed Setup for Deep Learning Architecture without Elastic Weight Consolidation*

Similar to the architectures of [23] and [22] , the proposed architecture consists of four parts: A dense layer, initialized with unsupervised pretrained weights, which is followed by several LSTM layers, a wide dense layer and the RUL output. The following subsection describes each part in detail.

1) Architecture

Initialization of the *Dense Layer* is motivated by the following: Labeled data is expensive to create and therefore more scarce than unlabeled data. To make use of unlabeled data we therefore follow [23] and [22] by using unsupervised pretraining to initialize the first part of the proposed architecture.

Autoencoders (AE) are unsupervisedly trained feedforward neural networks, consisting of an encoder and decoder part as illustrated in Fig. 2. The network is trained to recover the input data at its output. While passing through the network, the data is compressed in the smaller *Code Layer*. Through the compression an internal lower dimensional representation of the data is created, which only contains the more important features of the data. In contrast to [23], we propose the usage of an additional sparsity constraint on the *Code Layer* in the form of a regularizing loss function that keeps the average activation of the *Code Layer* low to improve the feature extraction quality [33] of the encoder. This is called a sparse AE. Rectified Linear Units (ReLU) are used as activation function of the *Code Layer*. The pretraining process is done as follows: A sparse AE is unsupervisedly trained. Then the weights and biases of the encoder part are copied into the first dense layer.

The *Code Layer* is followed by *LSTM Layers*. Literature results in Table I suggest that LSTM based approaches outperform CNN and Deep Believe Network based approaches, especially on the PHME of datasets FD2 and FD4. Additionally, CNNs and Deep Believe Networks rely on fixed time windows,

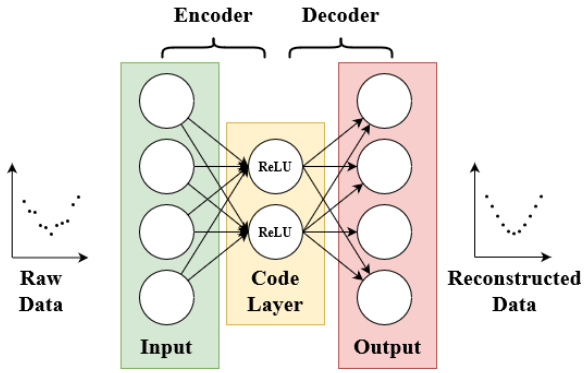

Fig. 2. Schematics of an autoencoder

which lessens their applicability. Therefore, this study utilizes LSTM.

An LSTM is a recurrent neural network architecture specialized on sequence data that can actively learn to remember information over a sequence. We propose the usage of the LSTM version by [34]. The main difference to the first described LSTM is its ability to also learn to actively forget information. Furthermore, choosing [34]'s LSTM version enables us to use a hardware accelerated implementation and thereby reduce the training time of our algorithm.

After the *LSTM Layers*, the high dimensional data is reduced with an additional fully connected layer, the *RUL Layer*. Its activation function is optimized by the hyperparameter search. Output of the proposed setup without EWC is the one-dimensional RUL value. Therefore, the last single neuron has a linear activation function.

Fig. 3 shows an overview of the proposed setup. The transparencies in each part of the setup symbolize hyperparameters that are chosen separately for each dataset by a hyperparameter search. Every layer except the first layer uses dropout, which is a regularization technique, first described by [35], that helps to prevent deep learning models from overfitting by randomly disabling neurons during the training process.

2) Hyperparameter search

The hyperparameter search to find a well performing architecture is carried out for each dataset individually. 20% of the training data is chosen randomly as validation data. To conduct the search, Tree of Parzen estimators as suggested by [36] and their hyperopt framework are used. Table II shows all hyperparameters and their respective search spaces. *Choice* means that the search space is constrained to the discrete list of values listed. *Uniform* indicates a continuous interval between the given values. *Data dim* is the dimension of the input data. Table III shows the results of the hyperparameter search for each turbofan dataset.

B. *Proposed Setup for Elastic Weight Consolidation*

In order to enhance the conventional deep learning architecture described so far to facilitate EWC, the diagonal of the Fisher matrix is needed (see subsection 2.B). The Fisher matrix for a task A is defined as the covariance matrix of the score function as defined by (4):

$$s_A(\theta) = \frac{\partial \log(L(X_A,\theta))}{\partial \theta} \tag{4}$$

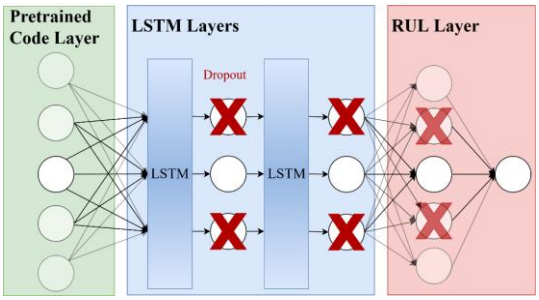

Fig. 3. Deep learning architecture used for the RUL prediction without elastic weight consolidation. Transparencies indicate variable layer sizes, which are hyperparameters.

TABLE II. HYPERPARAMETERS AND THEIR CORRESPONDING SEARCH SPACES. COLORS INDICATE LAYER ACCORDING TO FIG. 3.

Parameter Name	Search space
Code layer size	Choice (4; 7; ...; Input data dim)
# of LSTM layers	Choice (2, 3)
LSTM layer 1 size	Choice (25; 50; 100; 200; 300; 400)
Dropout 1	Uniform (0…1)
LSTM layer 2 size	Choice (25; 50; 100; 200; 300; 400)
Dropout 2	Uniform (0…1)
LSTM layer 3 size	Choice (25; 50; 100; 200; 300; 400)
Dropout 3	Uniform (0…1)
Dense layer size	Choice (25; 50; 100; 200; 300; 400)
Dense activation function	Choice (ReLU; sigmoid)
Dropout 4	Uniform (0…1)

TABLE III. RESULTS OF THE HYPERPARAMETER SEARCH FOR EACH TURBOFAN DATASET. COLORS INDICATE LAYER ACCORDING TO FIG. 3.

Parameter Name	FD1	FD2	FD3	FD4
Code layer size	7	19	13	7
# of LSTM layers	2	2	2	2
LSTM layer 1 size	100	200	50	100
Dropout 1	0.61	0.73	0.54	0.61
LSTM layer 2 size	100	200	50	100
Dropout 2	0.73	0.26	0.33	0.74
Dense layer size	100	100	25	100
Dense activation funct.	ReLU	sigmoid	ReLU	ReLU
Dropout 4	0.65	0.08	0.1	0.65

To be able to calculate the score, we require the prediction likelihood $L(X_A, \theta)$ of the model that is trained on a task A given some parameters θ on a data point X_A of that task. The setup for RUL prediction as described in subsection 3.A and as used in the studies listed in Table I outputs no such prediction likelihood.

Therefore, we propose a change of the prediction problem from regression to classification. This is achieved by exchanging the last single linear activated output neuron with three output neurons with a softmax activation. Each output neuron stands for a predicted class representing the state of health (SoH) of a turbofan engine. These states of health are defined as RUL intervals given in Table IV.

Due to these architectural changes we can calculate the score as follows:

1. Draw a random data point X_A from the dataset of task A.
2. Calculate the prediction probabilities vector p_A of the trained model on the drawn data point.
3. Choose a single class as prediction by sampling from a multinomial distribution with probabilities p_A. This sampling process is the likelihood function in the definition of the score function.
4. Calculate the partial derivatives as defined in the score function. The output is a mapping of how strongly the class prediction varies with each weight.

The described steps are repeated for many data points. For the turbofan dataset 500 data points are used. The diagonal

TABLE IV. DEFINITION OF THE CLASSES BY REMAINING USEFUL LIFE (RUL) VALUES AND STATE-OF-HEALTH (SOH) INTERPRETATION

Class No.	1	2	3
RUL Definition	< 60	60 … 129	≥ 130
SoH Interpretation	Critical	Declining	Normal

TABLE V. HYPERPARAMETERS USED FOR THE ELASTIC WEIGHT CONSOLIDATION EXPERIMENTS. COLORS INDICATE LAYER ACCORDING TO FIG. 3.

Parameter Name	Parameters used
Dense layer size	30
LSTM layer 1 size	200
LSTM layer 2 size	200
Dense layer size	3
Dense activation function	softmax

entries of the covariance matrix of all calculated scores give us the $F_{A,i}$ values for the EWC loss function. To validate our implementation of EWC we repeated [10]'s experiments on the MNIST dataset [37] successfully.

The SoH classification is easier to learn than the RUL regression, therefore no hyperparameter search is conducted and no dropout is used. Applying the unsupervised pre-training when EWC is used would add unknown quantities of prior knowledge into the network, we therefore do not use it for EWC experiments. Thus, the first layer is a normal dense layer. Table V shows the hyperparameter architecture used for the EWC experiments. The last fully connected layer is of size 3, because of the 3 SoH classes.

IV. EXPERIMENTS

In this section, the experiments carried out using the proposed deep learning architectures on the turbofan degradation datasets as well as their results are described. The non-EWC architecture's results are presented first, followed by the ones delivered by the EWC implementation. Certain areas in the diagrams presented are highlighted and numbered in order to facilitate better understanding.

A. Setup of the Experimental Study

All experiments were carried out on a computer with an Intel i7-7700K CPU and a NVIDIA GeForce GTX 1060 6GB GPU running Ubuntu 18.04.3 LTS. The learning framework used was Tensorflow 1.13.2.

The turbofan engine data was transformed into sub-sequences of length 50 and normalized between 0 and 1. The RUL values were created by a piecewise linear function with a

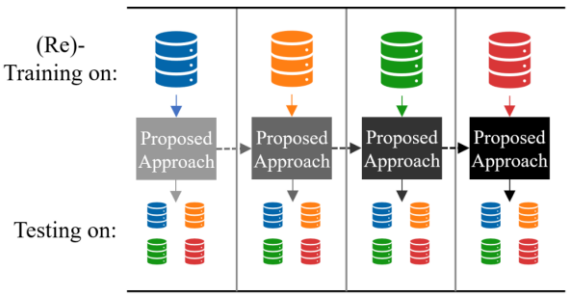

Fig. 4. Training and Testing sequence used in the experiments

start of the linear phase at cycle 130. To save training time, sensors whose signals are constant over the whole sequence were removed.

Each experiment is conducted as illustrated in Fig. 4: In four sequential training sessions, the algorithm is trained on one dataset each. After each training session, the algorithm is tested on all datasets.

B. *Results of the Deep Learning Architecture Experiments without Elastic Weight Consolidation*

Table VI shows the results achieved by the proposed semi-supervised approach on the four TEDSDS. Our proposed Approach is in second place behind [22] in all PHME metrics except for FD2, where it is third behind [21].

However, while [22]'s approach outperforms Proposed Approach on all metrics, it should be noted that their optimization technique takes about 60 hours on every dataset. The hyperparameter search we used takes between 2 and 4 hours using the same GPU.

Proposed Approach outperforms [17]'s CNN on every dataset measured by the PHME, but their RMSE is better each time. This means that Proposed Approach has fewer predictions that overestimate the RUL, which is important for a safety critical appliance like turbofan engines.

C. *Results of the Elastic Weight Consolidation Experiments*

Before remembering through EWC can be properly evaluated, we need to establish how pronounced catastrophic forgetting is for the TEDSDS. Fig. 5 shows the accuracy changes of a sequential training on the four TEDSDS without EWC. During the training of FD1 the accuracy of FD3 rises. FD2 and FD4 show a similar relation. The different datasets are therefore not independent. However, a clear downward trend of FD2's accuracies during training of FD3 can be seen at highlight 1 in Fig. 5. At highlight 2, the end of the training, FD1 and FD3's performances are low and FD4 and FD2 perform well due to their relatedness. Accordingly, forgetting of FD1 and FD3 is occurring.

Fig. 6 shows the same sequential training as Fig. 5 but with EWC. Comparing both figures at highlights 2 and 4 suggests: After the training, FD1 and FD4 are remembered significantly better than without EWC while the accuracies of FD2 and FD4 have slightly decreased through EWC. FD2's accuracy loss at highlight 1 is prevented by EWC at highlight 3. The much more pronounced downswing of FD3's accuracy during the second training phase might be caused by EWC, but is not problematic, because the model has not yet trained on FD3. While EWC improves the training somewhat, in total the performance of EWC is not satisfactory, because FD1 and FD3's end accuracies are much lower than after their individual training. Experiments with higher λ parameters were conducted to force remembering, but the training did not converge due to a sharp rise of loss values during the training of FD3.

TABLE VI. RESULTS OF THE PROPOSED APPROACH COMPARED TO LITERATURE RESULTS ON THE TEDSDS: PHME (UPPER VALUE) AND RMSE (LOWER VALUE)

Algorithm	Dataset			
	FD1	***FD2***	***FD3***	***FD4***
CNN [17]	274 12.61	10412 22.36	284 12.64	12466 23.31
Proposed Approach	259 13.03	4617 24.28	253 13.60	4367 26.04
Restricted Boltzmann Machine + LSTM [22]	231 12.56	3366 22.73	251 12.10	2840 22.66

This difference between EWC's performance on the randomly permutated MNIST datasets and the TEDSDS could be explained by two major differences between the datasets themselves:

1. All MNIST datasets are completely independent from each other. Therefore, learning one task does not increase accuracy on the others. This is clearly not the case for the TEDSDS, as can be seen in Fig. 5.
2. The sub-tasks are not of equal difficulty. While the MNIST sub-tasks are all equally hard to learn for a neural network (Kirkpatrick et al. 2017), the different TEDSDS vary in complexity. This can be seen in Table I and Table VI. Every model performs better on FD1 and FD3 than on FD2 and FD4. Fig. 5 shows that FD1 and FD3 are also learned faster, i.e. after fewer epochs, than FD2 and FD4.

To investigate if the differences in difficulty causes the reduced performance of EWC on the TEDSDS, another set of

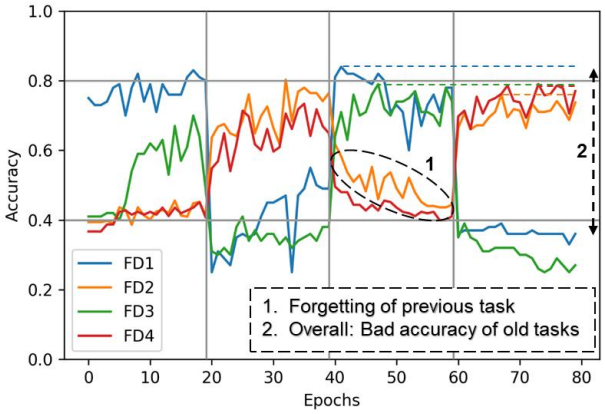

Fig. 5. Accuracy changes during sequential training of TEDSDS without EWC demonstrating catastrophic forgetting. Vertical lines mark the start of a new dataset. Training order: FD1, FD2, FD3, FD4.

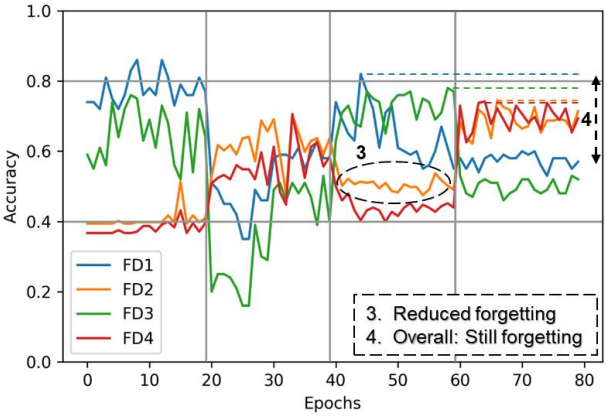

Fig. 6. Accuracy changes during sequential training of TEDSDS with EWC showing improved performance. Vertical lines mark the start of a new dataset. Training order: FD1, FD2, FD3, FD4.

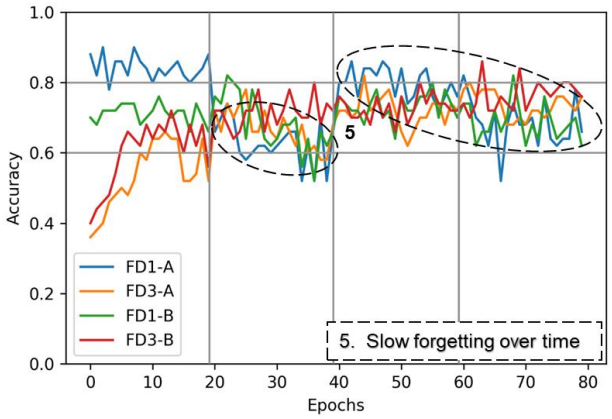

Fig. 7. Training without EWC on split datasets demonstrating catastrophic forgetting. Training order: FD1-A, FD3-A, FD1-B and FD3-B.

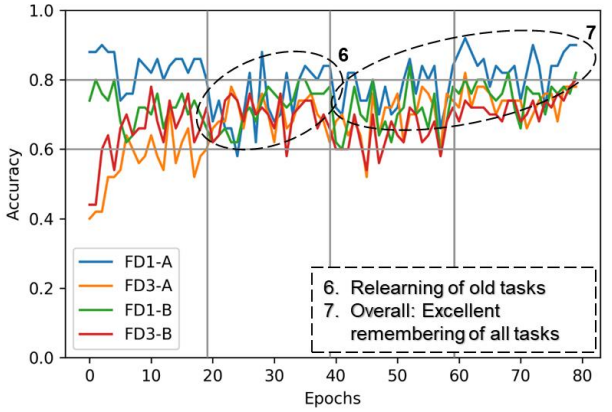

Fig. 8. Training with EWC on split datasets showing improved performance. Training order: FD1-A, FD3-A, FD1-B and FD3-B.

experiments was conducted. The two less complex datasets FD1 and FD3 are each split into two sub-datasets: FD1-A/-B and FD3-A/-B. First, a training without EWC is run, whose results can be seen in Fig. 7: A decline of the performance (forgetting) of FD1-A during the trainings of FD3-A and FD3-B occurs at highlight 5. During the training of FD1-B, the performance of FD1-A increases again, because both sub-datasets originate from FD1. However, at the end of the experiment, accuracies of FD1-A and FD1-B are significantly lower than during their training. FD3-A and FD3-B are from the same dataset, therefore FD3-A was not forgotten and both show a good performance.

The same training is run with EWC as shown in Fig. 8. Accuracy of FD1-A first declines at the beginning of the training on FD3-A, but then recovers in highlight 6. At the end of the training with EWC, the accuracies for all sub-datasets are close to their maximum value. We were also able to increase the EWC parameter λ from 13 to 45 without a diverging training.

These results suggest that EWC works when different tasks are related, but its performance declines when the tasks are of vastly different complexity.

V. CONCLUSION

In this paper, the prediction of the remaining useful life of industrial machinery using deep learning architectures was examined with a focus on transferring knowledge across decentral sub-datasets. The publicly available Turbofan Engine Degradation Simulation Data Set by NASA was used for evaluation.

Key result: The decentralized learning architecture developed in this paper is capable of effective learning on smaller, decentral datasets without the need for centralized (cloud) storage. It could therefore solve the problem that arises from conventional deep learning based fault prediction' dependency on a centralized accumulation of training data, which currently hinders its deployment.

As the chosen method of continual learning, i.e. elastic weight consolidation, requires a conventional deep learning architecture as a foundation, we undertook a survey of those first. This literature study revealed a great heterogeneity of previous approaches with a strong tendency towards the combination of several different methods. Based on this data, we chose an autoencoder and an LSTM for our proposed deep learning architecture.

Our evaluation of this conventional deep learning architecture revealed a performance close to, but not as good as the best-performing published approach. However, using a very similar hardware, we could reduce the optimization time by factor 15 to 30 compared to the aforementioned publication.

We then expanded our deep learning architecture in order to facilitate continual learning by elastic weight consolidation. A thorough evaluation of this approach revealed, that

1. continual learning profits from (sub-)datasets being not independent from one another, and
2. continual learning suffers from (sub-)datasets being of different complexity.

These results concerning the elastic weight consolidation methodology call for further research regarding their specific impact and extend.

However, even prior to such investigation, the applicability of elastic weight consolidation towards industrial use cases like fault prediction appears to be high, because

1. the accuracy of the approach proposed in this paper is high despite the dataset, i.e. the scenario, being far from optimal regarding the different tasks' complexity,
2. different real-life scenarios concerning the same component usually display a great relatedness regarding the data recorded (i.e. the datasets are not independent, see above), and
3. different real-life scenarios concerning the same component are usually of similar complexity regarding the data recorded (see above).

The authors therefore propose to examine the potential of elastic weight consolidation on a broader scale in other industrial automation scenarios suffering from small datasets.

REFERENCES

[1] H. Kagermann, "Change Through Digitization—Value Creation in the Age of Industry 4.0," in *Management of permanent change*, H. Albach, H. Meffert, A. Pinkwart, and R. Reichwald, Eds., Wiesbaden: Springer Fachmedien, 2015, pp. 23–45.

[2] X. Yao, J. Zhou, J. Zhang, and C. R. Boer, "From Intelligent Manufacturing to Smart Manufacturing for Industry 4.0 Driven by Next

Generation Artificial Intelligence and Further On," in *Industrial digitalization by enterprise systems: 2017 5th International Conference on Enterprise Systems (Proceedings)*, Beijing, 2017, pp. 311–318, DOI: 10.1109/ES.2017.58.

[3] B. Lindemann, C. Karadogan, N. Jazdi, M. Liewald, and M. Weyrich, "Cloud-based Control Approach in Discrete Manufacturing Using a Self-Learning Architecture," *IFAC-PapersOnLine*, vol. 51, no. 10, pp. 163–168, 2018, DOI: 10.1016/j.ifacol.2018.06.255.

[4] E. Lughofer and M. Sayed-Mouchaweh, "Prologue: Predictive Maintenance in Dynamic Systems," in *Predictive Maintenance in Dynamic Systems: Advanced Methods, Decision Support Tools and Real-World Applications*, E. Lughofer and M. Sayed-Mouchaweh, Eds., Cham: Springer International Publishing, 2019, pp. 1–23.

[5] S. Heo and J. H. Lee, "Fault detection and classification using artificial neural networks," *IFAC-PapersOnLine*, vol. 51, no. 18, pp. 470–475, 2018, DOI: 10.1016/j.ifacol.2018.09.380.

[6] R. Yang, M. Huang, Q. Lu, and M. Zhong, "Rotating Machinery Fault Diagnosis Using Long-short-term Memory Recurrent Neural Network," *IFAC-PapersOnLine*, vol. 51, no. 24, pp. 228–232, 2018, DOI: 10.1016/j.ifacol.2018.09.582.

[7] H. Tercan, A. Guajardo, and T. Meisen, "Industrial Transfer Learning: Boosting Machine Learning in Production," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN): Proceedings*, Helsinki, Finland, 2019, pp. 274–279, DOI: 10.1109/INDIN41052.2019.8972099.

[8] G. Xu *et al.*, "Data-Driven Fault Diagnostics and Prognostics for Predictive Maintenance: A Brief Overview," *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pp. 103–108, 2019, DOI: 10.1109/COASE.2019.8843068.

[9] B. Maschler, S. Kamm, N. Jazdi, and M. Weyrich, "Distributed Cooperative Deep Transfer Learning for Industrial Image Recognition," Preprint, 2020, DOI: 10.13140/RG.2.2.14189.41440/1.

[10] J. Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 114, no. 13, pp. 3521–3526, 2017, DOI: 10.1073/pnas.1611835114.

[11] P. Durdevic, D. Ortiz-Arroyo, S. Li, and Z. Yang, "Vision Aided Navigation of a Quad-Rotor for Autonomous Wind-Farm Inspection," *IFAC-PapersOnLine*, vol. 52, no. 8, pp. 61–66, 2019, DOI: 10.1016/j.ifacol.2019.08.049.

[12] B. Maschler and M. Weyrich, "Deep Transfer Learning at Runtime for Image Recognition in Industrial Automation Systems," *2020 16th Technical Congress EKA - Design of Complex Automation Systems, University of Magdeburg*, 2020.

[13] A. Saxena and K. Goebel, "Turbofan engine degradation simulation data set," *NASA Ames Prognostics Data Repository*, 2008.

[14] D. K. Frederick, J. A. DeCastro, and J. S. Litt, "User's guide for the commercial modular aero-propulsion system simulation (C-MAPSS)," 2007.

[15] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *International Conference on Prognostics and Health Management, 2008*, Denver, USA, 2008, pp. 1–9, DOI: 10.1109/PHM.2008.4711414.

[16] G. S. Babu, P. Zhao, and X.-L. Li, "Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life," in *21st International Conference on Database Systems for Advanced Applications DASFAA*, Dallas, USA, 2016, pp. 214–228, DOI: 10.1007/978-3-319-32025-0_14.

[17] X. Li, Q. Ding, and J.-Q. Sun, "Remaining useful life estimation in prognostics using deep convolution neural networks," *Reliability Engineering & System Safety*, vol. 172, pp. 1–11, 2018, DOI: 10.1016/j.ress.2017.11.021.

[18] C. Zhang, P. Lim, A. K. Qin, and K. C. Tan, "Multiobjective Deep Belief Networks Ensemble for Remaining Useful Life Estimation in Prognostics," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2306–2318, 2017, DOI: 10.1109/TNNLS.2016.2582798.

[19] J. Yang, X. Zeng, S. Zhong, and S. Wu, "Effective neural network ensemble approach for improving generalization performance," *IEEE transactions on neural networks and learning systems*, vol. 24, no. 6, pp. 878–887, 2013, DOI: 10.1109/TNNLS.2013.2246578.

[20] R. Coop, A. Mishtal, and I. Arel, "Ensemble learning in fixed expansion layer networks for mitigating catastrophic forgetting," *IEEE transactions on neural networks and learning systems*, vol. 24, no. 10, pp. 1623–1634, 2013, DOI: 10.1109/TNNLS.2013.2264952.

[21] S. Zheng, K. Ristovski, A. Farahat, and C. Gupta, "Long Short-Term Memory Network for Remaining Useful Life estimation," in *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, Dallas, TX, USA, 2017, pp. 88–95, DOI: 10.1109/ICPHM.2017.7998311.

[22] A. L. Ellefsen, E. Bjørlykhaug, V. Æsøy, S. Ushakov, and H. Zhang, "Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture," *Reliability Engineering & System Safety*, vol. 183, pp. 240–251, 2019, DOI: 10.1016/j.ress.2018.11.027.

[23] Y. Song, G. Shi, L. Chen, X. Huang, and T. Xia, "Remaining Useful Life Prediction of Turbofan Engine Using Hybrid Model Based on Autoencoder and Bidirectional Long Short-Term Memory," *J. Shanghai Jiaotong Univ. (Sci.)*, vol. 23, no. S1, pp. 85–94, 2018, DOI: 10.1007/s12204-018-2027-5.

[24] D. Maltoni and V. Lomonaco, "Continuous learning in single-incremental-task scenarios," *Neural networks : the official journal of the International Neural Network Society*, vol. 116, pp. 56–73, 2019, DOI: 10.1016/j.neunet.2019.03.010.

[25] R. French, "Catastrophic forgetting in connectionist networks," *Trends in Cognitive Sciences*, vol. 3, no. 4, pp. 128–135, 1999, DOI: 10.1016/S1364-6613(99)01294-2.

[26] H. J. Sussmann, "Uniqueness of the weights for minimal feedforward nets with a given input-output map," *Neural Networks*, vol. 5, no. 4, pp. 589–593, 1992, DOI: 10.1016/S0893-6080(05)80037-1.

[27] F. Huszár, "Note on the quadratic penalties in elastic weight consolidation," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 115, no. 11, E2496-E2497, 2018, DOI: 10.1073/pnas.1717042115.

[28] J. Kirkpatrick *et al.*, "Reply to Huszár: The elastic weight consolidation penalty is empirically valid," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 115, no. 11, E2498, 2018, DOI: 10.1073/pnas.1800157115.

[29] C. V. Nguyen, Y. Li, D. B. Thang, and R. E. Turner, "Variational continual learning," in *6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2018, DOI: 10.17863/CAM.35471.

[30] C. Baweja, B. Glocker, and K. Kamnitsas, "Towards continual learning in medical imaging," in *Medical Imaging meets NeurIPS (Workshop), on the 32nd Conference on Neural Information Processing Systems (NeurIPS)*, Montreal, 2018.

[31] K. van Garderen, S. van der Voort, F. Incekara, M. Smits, and S. Klein, "Towards continuous learning for glioma segmentation with elastic weight consolidation," 2019.

[32] B. Thompson, J. Gwinnup, H. Khayrallah, K. Duh, and P. Koehn, "Overcoming Catastrophic Forgetting During Domain Adaptation of Neural Machine Translation," in *Proceedings of the 2019 Conference of the North*, Minneapolis, Minnesota, pp. 2062–2068, DOI: 10.18653/v1/N19-1209.

[33] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Cambridge, Massachusetts, London, England: MIT Press, 2016.

[34] F. A. Gers, "Learning to forget: continual prediction with LSTM," in *ICANN99: Ninth International Conference on Artificial Neural Networks*, Edinburgh, UK, 1999, pp. 850–855, DOI: 10.1049/cp:19991218.

[35] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[36] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," 2013.

[37] Y. LeCun, C. Cortes, and C. J.C. Burges, *The MNIST database of handwritten digits*. Dataset. Accessed on: Mar. 27 2020.