Original Article

Institution of MECHANICAL ENGINEERS
Journal of RISK AND RELIABILITY

Proc IMechE Part O:
J Risk and Reliability
2020, Vol. 234(1) 151–167

DOI: 10.1177/1748006X19866546
journals.sagepub.com/home/pio
SAGE

A novel deep capsule neural network for remaining useful life estimation

Andrés Ruiz-Tagle Palazuelos[1], Enrique López Droguett[1,2] and Rodrigo Pascual[3]

Abstract

With the availability of cheaper multi-sensor systems, one has access to massive and multi-dimensional sensor data for fault diagnostics and prognostics. However, from a time, engineering and computational perspective, it is often cost prohibitive to manually extract useful features and to label all the data. To address these challenges, deep learning techniques have been used in the recent years. Within these, convolutional neural networks have shown remarkable performance in fault diagnostics and prognostics. However, this model present limitations from a prognostics and health management perspective: to improve its feature extraction generalization capabilities and reduce computation time, ill-based pooling operations are employed, which require sub-sampling of the data, thus loosing potentially valuable information regarding an asset's degradation process. Capsule neural networks have been recently proposed to address these problems with strong results in computer vision–related classification tasks. This has motivated us to extend capsule neural networks for fault prognostics and, in particular, remaining useful life estimation. The proposed model, architecture and algorithm are tested and compared to other state-of-the art deep learning models on the benchmark Commercial Modular Aero Propulsion System Simulation turbofans data set. The results indicate that the proposed capsule neural networks are a promising approach for remaining useful life prognostics from multi-dimensional sensor data.

Keywords

Prognostics and health management, remaining useful life, deep learning, convolutional neural network, capsule neural network, reliability

Date received: 4 February 2019; accepted: 25 June 2019

Introduction

Nowadays, in the context of reliability and maintenance engineering as well as in prognostics and health management (PHM), sensing technologies are getting higher attention due to the increasing need to accurately predict the future behavior of degradation processes in complex physical assets such as compressors and turbines.[1] PHM aims to make diagnosis of asset's health status based on multi-sensory monitoring data, predicting behavioral anomalies and executing suitable maintenance actions before problems take place. In this context, the estimation of remaining useful life (RUL) has become a crucial task in PHM, where the goal is to increase the asset's operational availability and reduce maintenance costs associated with unnecessary interventions.[2] This is not an easy task for a data-driven model since, in many real-world industrial applications, it is usually not possible to precisely determine the asset's health condition and its associated RUL at each time step without an accurate physics-based model.

Generally, PHM techniques can be categorized into three types of approaches: model-based approaches,[3] data-driven approaches[4] and hybrid approaches.[5] Model-based approaches incorporate a physical background of the asset's behavior and degradation, but require a large amount of prior knowledge about its failure mechanisms, thus becoming ineffective if these are unknown. However, data-driven approaches use only historical sensor data to infer and learn meaningful features to indicate an asset's health condition like

[1]Department of Mechanical Engineering, University of Chile, Santiago, Chile
[2]Center for Risk and Reliability, University of Maryland, College Park, MD, USA
[3]Mechanical Engineering Department, University of Concepción, Concepción, Chile

Corresponding author:
Enrique López Droguett, Department of Mechanical Engineering, University of Chile, Av Beauchef 851, Santiago 70000, Chile.
Email: enlopez@uchile.cl

its damage state and RUL, usually making them easier to be generalized. Therefore, various data-driven frameworks and models have been proposed in recent years to tackle prognostics problems such as the ones based on support vector machines (SVMs),[6] artificial neural networks (ANNs)[7] and, most recently, deep learning networks,[8] with the latter delivering results that outperform the ones from the shallow models (i.e. SVMs and ANNs).[8] Hybrid approaches combine the advantages of the previous two, but, to this day, it has been very challenging to implement them.

Among the underlying reasons for the above-mentioned success of deep learning networks based models are their capacity and flexibility in dealing with highly non-linear multi-dimensional sensor data, learning mappings between the features of the gathered data to a physical asset's RUL. These models are characterized as neural networks with deep architectures, where many layers of neurons are stacked together to obtain high-level abstractions of the data. For example, Tian[9] proposed a deep ANN to estimate the RUL of pump bearings using in-field inspection monitoring data. Fink et al.[10] developed a deep feedforward neural network based on multi-values neurons for predicting the degradation of railway track turnouts using time-series real data in a turbocharger benchmark time-to-failure data set. Ren et al.[11] proposed a fully connected deep neural network to estimate the RUL of rolling bearings using the FEMTO-ST PRONOSTIA data set[12] by collecting features directly from raw-output sensor data. Zheng et al.[13] implemented a deep long short-term memory recurrent neural network (DLSTM) for RUL estimation in turbofans under complex operation modes and hybrid degradations. A multi-objective deep belief network (MODBNE), based on an evolutionary ensemble method, was proposed by Zhang et al.[14] that involves multiple deep belief networks (DBNs) simultaneously subject to diversity and accuracy as conflicted objectives that are combined to establish a model for RUL estimation, evaluating its performance on NASA's Commercial Modular Aero Propulsion System Simulation (C-MAPSS) turbofans time-to-failure data set.[15]

Moreover, convolutional neural networks (CNNs) have become the dominant technique among deep learning architectures to handle computer vision (CV) and image classification problems,[16–21] in some cases overcoming human accuracy.[22,23] Since CNNs can automatically collect high-level abstraction features from raw data, they have recently been used in fault diagnostics. Indeed, on machinery fault diagnosis, Chen et al.[24] used a shallow CNN architecture for gearbox vibration data, obtaining a better classification performance compared to SVMs. Wang et al.[25] used wavelet scalogram images as an input to a deep CNN (DCNN) consisting of convolutional and sub-sampling layers (i.e. pooling layers for feature dimensionality reduction) for rolling bearing fault diagnosis. Guo et al.[26] implemented an adaptive DCNN for the Case Western's bearing data set[27] to perform fault diagnosis. Verstraete et al.[28] proposed a DCNN architecture for fault identification and classification on rolling bearings in the context of the MFPT[29] and Case Western's data sets with minimal data preprocessing, outperforming other deep learning–based methods and showing robustness against experimental noise. Modarres et al.[30] proposed a DCNN for detection and identification of structural damage, obtaining strong results on honeycombs structures and concrete bridge crack identification.

Although CNNs are designed for CV and classification, their applicability to fault prognostics problems have also been explored. Indeed, NASA's C-MAPSS turbofans time-to-failure data set have been extensively analyzed[15] with RUL estimation as the main focus. Sateesh Babu et al.[31] proposed the first CNN-based model for RUL prediction where convolution and pooling filters were applied along the temporal dimension of data. Also, Li et al.[32] proposed a DCNN where a time-window approach is employed as preprocessing of the turbofans C-MAPSS data set and convolution filters were applied along the temporal dimension.

Even though CNNs have displayed strong performance in automatic feature extraction, they do present some weaknesses. Indeed, as stated by Hinton et al.,[33] CNNs are misguided in what they are trying to achieve: they do not aim to a "viewpoint invariant" (i.e. equivariant; this means that the network will not lose accuracy if changes are made in the test set, such as alterations in rotation and skewness among others) feature extraction using "neurons" that use a single scalar output to summarize the activities of the convolutional and pooling layers. This is also perceived in fault diagnosis and prognosis. In fact, a common practice for RUL estimation with CNNs is to apply the convolution and pooling filters along the temporal dimension of sensoring input data.[31,32] This might lead to loss of information and subsequent poor fault diagnosis and prognosis performance as some important features could be lost, such as periodicity and low-valued amplitudes in signals (e.g. vibration and acoustic emission monitoring).[34] The main underlying reason is that, using the pooling operation in CNNs, "neurons" in one layer are allowed to ignore all but the most active feature detector in local pool from a previous convolutional layer and thus throwing away information about its precise position.

As an attempt to address these difficulties in classification-related problems, Hinton et al.[33] proposed that neural networks should use "capsules of neurons" that encapsulate the extracted features by the convolution operation in a vector-shaped highly informative output with the goal of achieving a better hierarchical feature extraction of the input data, without losing relevant positional information in the process. This new concept was introduced by Sabour et al.[35] as a new neural network architecture known as capsule neural networks (CapsNets), achieving state-of-the-art

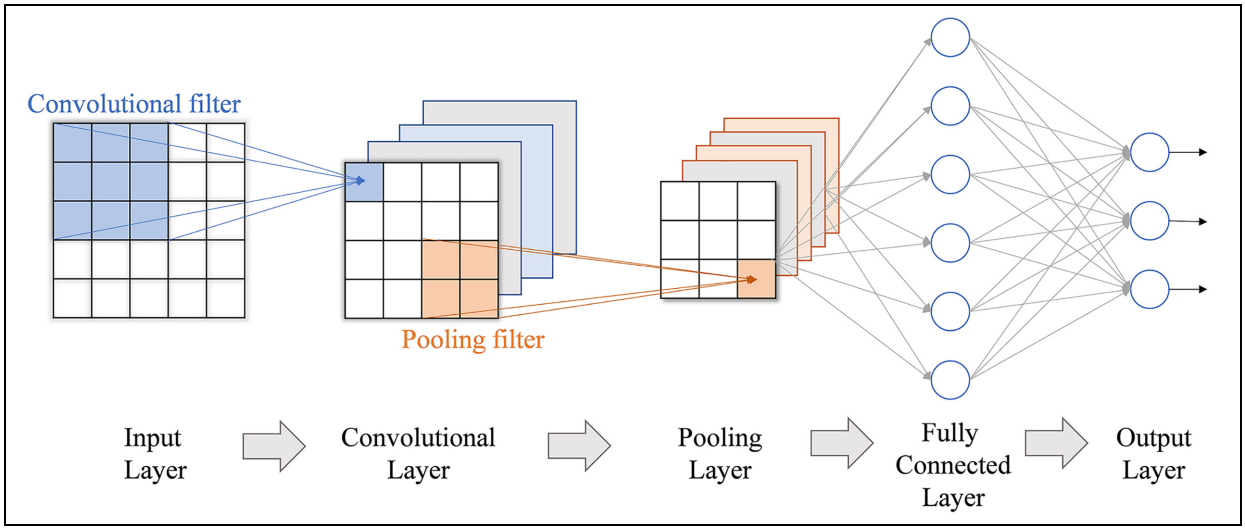

Figure 1. Generic two-dimensional CNN architecture.

results in CV-related classification tasks involving the MNIST and Multi-MNIST data sets.[36] In the context of PHM, Zhu et al.[37] applied an inception-based CapsNets model as image-based data feature extractor (as commonly done with CNNs) to Case Western and Paderborn University[38] data sets to perform bearing fault diagnostics, outperforming CNN-based models in this task. Furthermore, Zhu et al. proposed a regression output branch for the model used in Case Western data set, with the aim of estimating the damage size in the bearing, which was limited to a discrete assessment and therefore it could be of only three possible values. Although this is a first approximation for performing regression with CapsNets, having only three possible target damage sizes values can be redundant to separate from a classification model, being the continuous target data regression a much more interesting and useful problem to solve from a PHM perspective.

The above-mentioned perceived weakness of CNNs and the results obtained for fault classification tasks by Zhu et. al. with the use of CapsNets have motivated us to extend CapsNets' capabilities in classification for which they were made (i.e. discrete diagnostics tasks) to regression problems in general (i.e. using continuous data estimation as a target), which, in the context of PHM, is achieved by putting forward a CapsNet model, architecture and algorithm for RUL prognostics. To the best of the authors' knowledge, this is the first time a CapsNet is proposed for RUL estimation. To tackle this challenge, the proposed model's architecture consists of two convolutional layers for feature extraction from sensor signals (i.e. the spatial domain) and then the feature maps are reshaped as an array of primary high-informative capsule layers designed to contain low-hierarchy temporal features. These low-level capsules are connected to a secondary capsules layer designed to detect and extract more complex high-hierarchy temporal features from the signal data. These last set of features are then fed to two consecutives fully connected multi-layer perceptron (MLP) layers for RUL prediction. The proposed model is exemplified and validated by means of the C-MAPSS turbofans data set and compared against other deep learning models such as CNNs and long short-term memories (LSTMs).

The remaining of this article is structured as follows: section "CNNs and their limitations" gives a brief background on CNNs and also discusses their main limitations and drawbacks. Section "CapsNets for RUL estimation" introduces and discusses the proposed CapsNet model, architecture and algorithm for RUL estimation. In section "Case study: C-MAPSS turbofans," the effectiveness of the proposed model is analyzed by means of the C-MAPSS turbofans data set. The final section provides some concluding remarks.

CNNs and their limitations

Among deep learning architectures, CNNs have been applied to the learning of mappings between features of gathered sensory data and its related diagnosis or prognosis metric. To do this, CNNs were inspired by the visual cortex,[39] where a series of layers of neurons are connected to model hierarchical abstractions of data to interpret more complex representations, performing automatic feature extraction using two types of operations: convolution and pooling.

A generic two-dimensional (2D) CNN is shown in Figure 1, where the input layer is the preprocessed data to be analyzed as a 2D array (i.e. a spectrogram). Then, a set of convolutional filters (i.e. kernels) is applied, creating a stack of feature maps. As shown in equation (1), these filters consist of a locally summed set of weights, which are subsequently passed through to a non-linear

activation function. A rectified linear unit (ReLU) activation function, shown in equation (2), is commonly used due to its faster training time.[40] Thus

$$\boldsymbol{x}_l^{(k)} = \phi\left(\sum_{c=1}^{C} \boldsymbol{w}_l^{(c,k)} * \boldsymbol{x}_{l-1}^{(c)} + \boldsymbol{b}_l^{(k)}\right) \quad (1)$$

$$\mathrm{ReLU}(x) = \max(x, 0) \quad (2)$$

where $*$ represents the convolution operator; $\boldsymbol{x}_l^{(k)}$ is the output of a neuron in the convolutional layer l and feature map k; ϕ is the non-linear activation function; $\boldsymbol{w}_l^{(c,k)}$ is the filter weight matrix of any neuron in the feature map k and convolutional layer l associated with its input in feature map c in the layer $(l-1)$; and $\boldsymbol{b}_l^{(k)}$ is the bias term for the feature map k in the convolutional layer l.

After the convolutional layer, a pooling layer is applied over its output. This operation has three purposes: dimensionality reduction, translational invariance and, subsequently, computational cost reduction. Dimensionality reduction is achieved by replacing the output of the convolutional layer at a certain location by a statistic summary of the neighboring outputs. Consequently, the output of the pooling layer becomes approximately invariant to small translations of the input.[41] Without pooling, CNNs would only fit for data which are very close to the training set. Two typical pooling operations are max-pooling (MaxPool) and average-pooling (AvgPool), where a pooling filter is applied over the convolutional layer output, keeping only the highest value and the filter-average value, respectively. Finally, the output of the pooling layer is fed into a fully connected MLP that performs the fault diagnosis (usually simple classification) or prognosis (regression) tasks.

As discussed before, the pooling operation is a very important component in a CNN architecture as it introduces translational invariance that reduces the model's overfitting and thus being less susceptible to accuracy reduction when the test set is not close to the training set.[41] However, as stated before, translational invariance is not enough for a CNN to extract useful features for fault diagnosis and prognosis, and one should make equivariance as a new goal. Although translational invariance makes the CNN less sensitive to small changes in the test set, equivariance makes the CNN "understand" these changes and adapt itself accordingly so that the spatial location of a local extracted feature is not lost, thus positional hierarchy relationships between low- and high-level features are assured.[33]

This is a key feature in the context of PHM since engineers usually do not have prior knowledge about the features that are automatically extracted by the convolution and pooling operations and their relevance to map the relationships between the sensor data and their related classification or regression metrics.[28] Thus, when a pooling layer is employed in CNNs, important information could be lost. For example, Figure 2 shows a normalized sub-sample of a random signal from vibration sensor data in the ball bearings MFPT data set[29] along with the MaxPool and AvgPool one-dimensional filters of size four applied to the signal. Note how some information is lost, such as consecutive picks and low-valued amplitudes, thus reducing the model's capability to extract potentially relevant features for PHM-related tasks, such as RUL estimation.

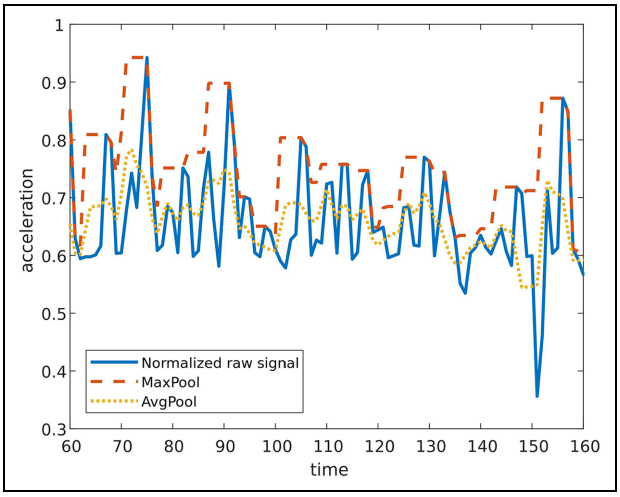

Figure 2. MFPT normalized raw signal with MaxPool and AvgPool operations applied.

To address the above weakness and limitations of CNNs in PHM tasks, the next section introduces a novel CapsNet architecture developed for regression problems, such as RUL prognostics.

CapsNets for RUL estimation

The idea of adding capsules of neurons in a CNN structure to enhance its feature extraction capabilities was first introduced in Hinton et al.[33] This was inspired by the cortical microcolumns present in the cerebral cortex for human vision, where low-level microcolumns store low-hierarchy representations of an object and are connected to high-level microcolumns, which in turn store high-hierarchy representations of the same object, allowing its identification. Then, when a new object is seen, the cerebral cortex deconstructs a hierarchical representation of it and tries to match it with already learned features present in cortical microcolumn connections.[42]

In the following sections, CapsNets are introduced, being the first neural networks that use capsules, however only for addressing classification problems. Then, the proposed CapsNet for RUL estimation is presented.

CapsNets

As mentioned before, CNNs use convolutional filters to extract features from an image, where, as seen in equation (1), high-level features combine low-level

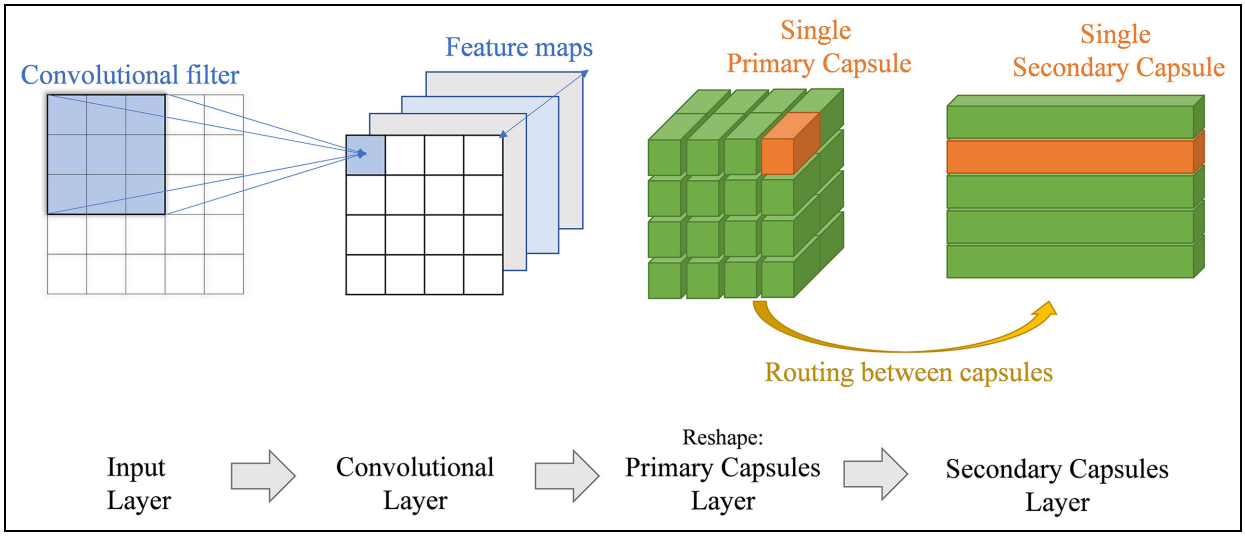

Figure 3. Basic CapsNet architecture.

features as a weighted sum. But this operation has a weakness: equivariance of information through the network is not necessarily achieved, so positional hierarchy relationships in the high-level features is not assured. This problem, as mentioned in section "CNNs and their limitations," is partially addressed by ill-based pooling layer operations that only aim for translational invariance. These difficulties are tackled by Hinton and colleagues[33,35,43] by means of CapsNets, which replace scalar-based outputs from "neurons" in CNNs by vector-based outputs from "capsules of neurons" to encapsulate highly informative hierarchical features that can be used for classification tasks.

As shown in Figure 3, a basic CapsNet architecture is composed by the following layers: input layer, convolutional layer, primary capsules layer and secondary capsules layer. The convolutional layer is applied in the same way as in CNNs, creating an output of feature maps of the input layer applying a convolutional filter.

To understand the other layers, two concepts must be introduced: capsules and routing between capsules. A capsule can be thought as a set of neurons whose main objective is to summarize features in a set of high-dimensional vectors of information. Therefore, a primary capsule layer is the output of a convolutional layer reshaped as a set of capsules representing a low positional hierarchy part of the input (see Figure 3). Then, these primary capsules are connected to a secondary capsule layer, composed by high-dimensional capsules that represent high positional hierarchy features of the input layer.

To obtain the probability of existence of a feature detected by a capsule, its norm is used. This is accomplished by a non-linearity named "squash"[35] that simply "squashes" a vector norm between 0 and 1, without losing any positional information in it, aiming for equivariance of information. This is presented in equation (3)

$$\text{squash}(\boldsymbol{a}) = \frac{||\boldsymbol{a}||^2}{1 + ||\boldsymbol{a}||^2} \frac{\boldsymbol{a}}{||\boldsymbol{a}||} \tag{3}$$

where $\boldsymbol{a}$ is a vector.

To make the connections between primary and secondary capsules, a trainable routing process between them is applied. In essence, a low-level capsule sends its input to a high-level capsule that judges if "agrees" with its input; if that is the case, both capsules are connected. To learn the best connections between capsules, the model runs through an iterative agreement process known as dynamic routing between capsules,[35] composed by the following steps:

1. For all capsule i in layer l and capsule j in layer $(l + 1)$:
   (a) A coefficient $b_{i,j}$ is initialized at zero.
   (b) A prediction vector $\hat{\boldsymbol{u}}_{j|i} = \boldsymbol{W}_{i,j}\boldsymbol{u}_i$ is created, where $\boldsymbol{u}_i$ is the output of capsule i in layer l and $\boldsymbol{W}_{i,j}$ is a weight matrix between this capsule and capsule j in layer $(l + 1)$.
2. For r rounds of routing by agreement, do:
   (a) For all capsule i in layer l, a couple coefficient is created: $\boldsymbol{c}_{i,j} = \text{softmax}(b_{i,j}) = \exp(b_{i,j}) / \sum_k \exp(b_{i,k})$.
   (b) For all capsule j in layer $(l + 1)$ a weighted sum $\boldsymbol{s}_j$ over all prediction vectors $\hat{\boldsymbol{u}}_{j|i}$ is created: $\boldsymbol{s}_j = \sum_i c_{i,j}\hat{\boldsymbol{u}}_{j|i}$.
   (c) For all capsule j in layer $(l + 1)$ apply the squash non-linearity, obtaining its output vector $\boldsymbol{v}_j = \text{squash}(\boldsymbol{s}_j)$.
   (d) For all capsule i in layer l and capsule j in layer $(l+1)$, an agreement metric $a_{i,j} = \hat{\boldsymbol{u}}_{j|i} \cdot \boldsymbol{v}_j$ is used to actualize $b_{i,j}^{(r+1)} = b_{i,j}^{(r)} + a_{i,j}$.
3. Return the output $\boldsymbol{v}_j$ from capsule j in layer $(l + 1)$.

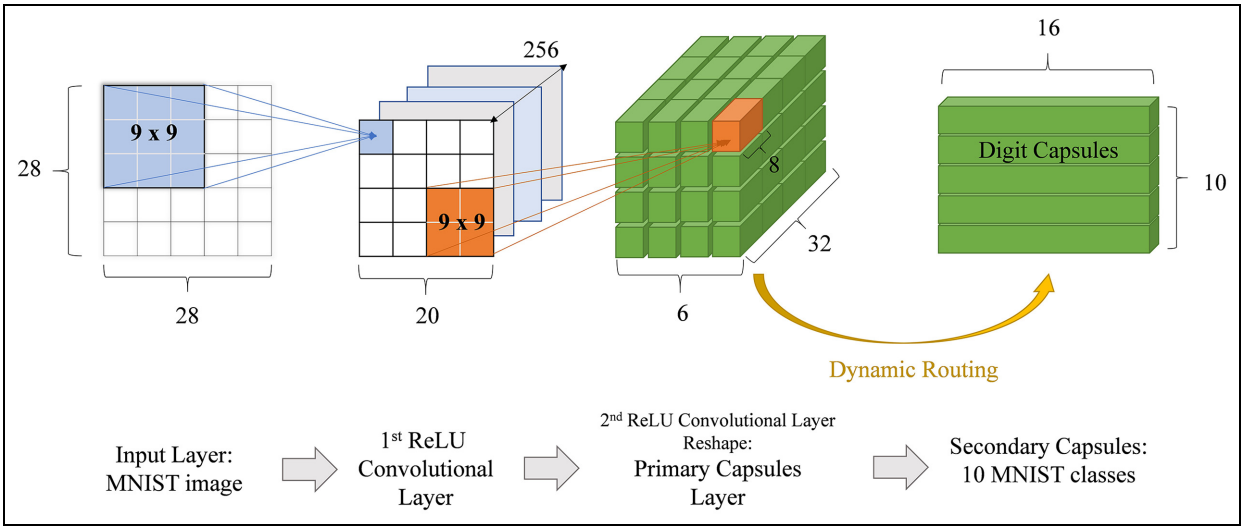

Figure 4. First CapsNet architecture for MNIST and multi-MNIST classification problems proposed in Sabour et al.[35]

Therefore, the dynamic routing process replaces the pooling operation and the way of connecting neurons through a neural network, learning to encode intrinsic spatial relationships between a part and a whole through the depth of the network equivariance of information between connected capsules,[35] achieving the following advantages over CNNs:

1. Equivariance ensures that lower to higher positional hierarchy features connections are reached deeper in the network layers, improving the feature generalization capability of the model.
2. Sub-sampling for dimensionality reduction (i.e. pooling operation) is not needed, so all the data is used to automatically extract relevant features for the aimed classification or regression tasks, without losing any relevant information.

The first CapsNet architecture, as shown in Figure 4, was proposed by Sabour et al.[35] for classification problems, evaluating its effectiveness in the MNIST and Multi-MNIST data sets. Here, the secondary capsules layer is named as "digit capsules," where each capsule in this layer represents a particular class (or digit in the case of MNIST). As before, the norm of each digit capsule represents the probability of existence of each class in the input data.

Proposed CapsNets for RUL estimation

The proposed CapsNet for RUL estimation focuses on automatic feature extraction from raw input sequential sensor signals, without the invariance and positional information loss problems associated with traditional CNN architectures. Assuming that the input data are normalized and preprocessed as a 2D sample from the raw signals,[14,32] with dimensions $N_t \times N_s$, where N_t denotes the temporal dimension and N_s the raw sensor signals (i.e. spatial domain), the proposed CapsNet for automatic feature extraction and RUL estimation is presented in Figure 5.

As seen in Figure 5, the idea behind the proposed CapsNet is to first process the input data in the sensory dimension (i.e. spatial domain) by two convolutional layers, obtaining high-level relationships between sensors. Then, in each time step, two layers of capsules connect these sensory relationships from the primary capsules to high-hierarchical relationships in the temporal domain contained in the secondary capsules. Finally, this information is passed to a two-layer fully connected MLP architecture for the RUL estimation (i.e. regression).

Note that, to extract relevant features from the raw signals in each single time step, a convolutional layer (Conv1) is applied over the spatial domain using filters of size 1×2 with stride $s = [1, 2]$, creating 32 feature maps of relationships between each couple of consecutive sensors. Assuming N_s as an even number, this operation reduces the dimension of the input sample to $N_t \times [N'_s = N_s/2]$, where N'_s denotes a new feature representation of the raw sensor signals in the spatial domain. Then, a second convolutional layer (Conv2) is applied over the spatial domain, using filters of size $1 \times N'_s$, relating all the features extracted by the first convolutional layer in each time step and giving as output a set of 32 feature maps of size $N_t \times 1$.

Next, the feature maps of the second convolutional layer are reshaped as a primary capsules layer (Primary Caps) consisting of $N_t \times 1 \times 32$ four-dimensional capsules that contain the spatial domain features and having four different capsules for each time step. The purpose of this is to let the capsules to detect and extract relevant low positional hierarchy features only from the temporal dimension. Then, these capsules are connected to a secondary capsules layer (Secondary Caps), where two eight-dimensional capsules are used

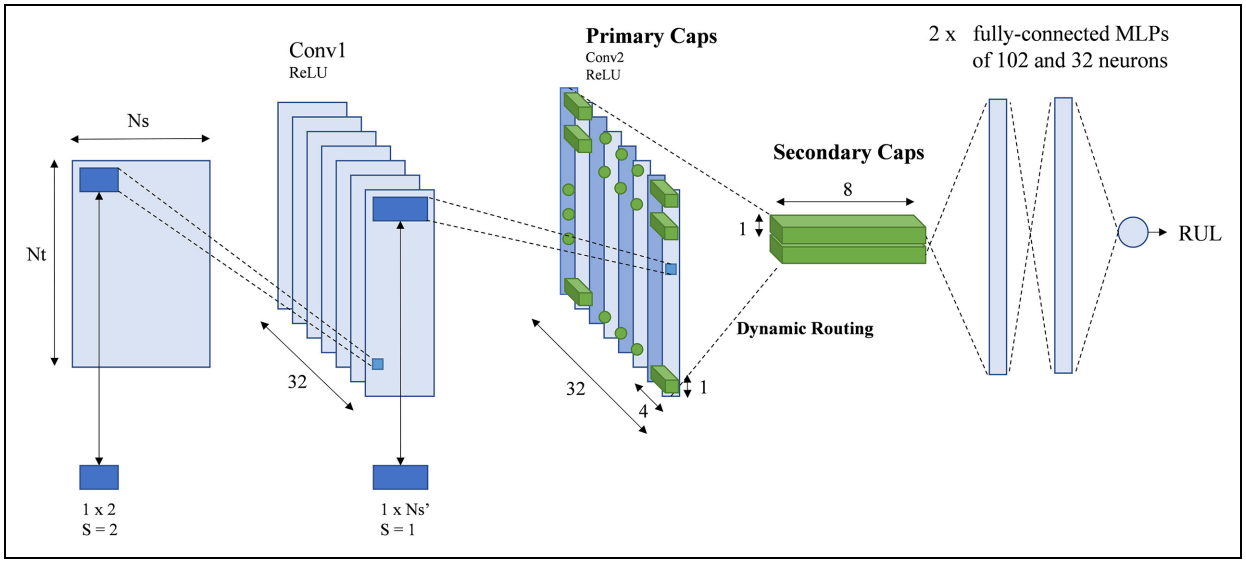

Figure 5. Proposed CapsNets architecture for RUL estimation. In the primary and secondary caps layers, the capsules are marked in green.

to extract high positional hierarchy features from the primary capsules. Two rounds of dynamic routing, as shown in section "CapsNet," are used to train (i.e. optimize) the connections between both capsules' layers. Finally, high temporal hierarchy features from the secondary capsules layer are flattened and passed to two fully connected layers with 102 and 32 neurons each, which carry out the regression for RUL estimation. Both convolutional layers and fully connected layers use ReLU as activation function. Figure 6 shows the proposed algorithm to implement the CapsNets for RUL estimation shown in Figure 5.

To optimize the proposed CapsNet model, the root mean squared error (RMSE) is used as loss function. To calculate it, the RUL of the training data set is used as the label to compare. This is presented in equation (4)

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_p - y_t\right)_i^2} \tag{4}$$

where N is the number of samples in the training set, y_p is the predicted RUL and y_t is the targeted true RUL.

In summary, the flowchart for RUL estimation via the proposed CapsNet model is shown in Figure 7. To train the model, a series of epochs are needed. The training set is divided in randomly generated batches (i.e. unrepeated sub-samples), so when all the batches pass through the network, an epoch is complete.

Case study: C-MAPSS turbofans

C-MAPSS data sets

In this section, the proposed CapsNet model for RUL prognosis is tested on the C-MAPSS benchmark data sets developed by NASA found at: https://data.nasa.gov/dataset/C-MAPSS-Aircraft-Engine-Simulator-Data/xaut-bemq/data.[15] C-MAPSS is divided in four data sets, FD001, FD002, FD003 and FD004, each one with corresponding training and testing sets that represent different simulations of a defined number of trajectories performed by turbofans. A summary of the data set is presented in Table 1. The time series corresponds to 21 raw sensor measurements that change from an undefined initial condition to an end point. For the training subset, this end point corresponds to a failure threshold, whereas for the test samples, it corresponds to an early stop before reaching the failure threshold. The objective is to estimate the RUL of each engine sample present in the testing subset. The actual RUL of each test sample is provided in the data set for validation.

Data preprocessing

Since the RUL labels for the training set are not provided, the approach presented in Peel[44] was employed to generate them. Indeed, even though the RUL target function can be represented as a linear degradation of an asset along its cycles of operation (in this case, turbofans), degradation can be assumed negligible in early stages of use for the turbofans under consideration. For the C-MAPSS data sets, this has been addressed by using a piecewise linear degradation behavior, where a number of operation cycles without the presence of degradation (R_{early}), with $R_{early} \in [120, 130]$, is used as maximum limit for the RUL labels in each data set. This threshold was proposed by Heimes[45] to preprocess the C-MAPSS data sets, finding that almost no degradation is present on test-engine samples running for less than 120 operation cycles. In this article, an

1. For each sensor in N_s do:
   - **1.1** Relate groups of sensors, applying a convolutional filter only in the spatial domain. As an output, a group of feature maps is created containing low-level relationships between sensor signals for each time step in N_t. This output is denoted as $\boldsymbol{x}_{sensor\_1}^{(k)}$ (see equation (1)).
   - **1.2** Apply a convolutional filter on each feature map k in $\boldsymbol{x}_{sensor\_1}^{(k)}$ only in the spatial domain, reducing its dimension to 1. As an output, a group of feature maps of shape $N_t \times 1$ is created containing high-level relationships between sensor signals for each time step in N_t. This output is denoted as $\boldsymbol{x}_{sensor\_2}^{(k)}$ (see equation (1)).
   - **1.3** Return $\boldsymbol{x}_{sensor\_2}^{(k)}$.
2. For each time step in N_t do:
   - **2.1** Reshape $\boldsymbol{x}_{sensor\_2}^{(k)}$ as a Primary Capsules layer. A capsule is created in each time step, containing in its dimensions a subset of feature maps from $\boldsymbol{x}_{sensor\_2}^{(k)}$. Primary capsules will represent low-hierarchy spatial relationships in time for high level sensory representations (extracted by the previous convolutions), denoted by $\boldsymbol{u}_{temporal\_1}$.
   - **2.2** Create a Secondary Capsules layer with less but more informative capsules than the previous layer. Each capsule in this layer will contain a set of high-hierarchy spatial relationships in the temporal dimension. Primary capsules in each time step will be connected to the secondary capsules by a process of dynamic routing of r rounds, giving as an output $\boldsymbol{v}_{temporal\_2}$.
   - **2.3** Return $\boldsymbol{v}_{temporal\_2}$.
3. Flatten the vectors (capsules) in $\boldsymbol{v}_{temporal\_2}$ and pass the information to a shallow fully-connected MLP based architecture for regression, obtaining the RUL estimation as an output.
4. Return RUL.

Figure 6. Proposed algorithm for feature extraction and RUL estimation.

Table 1. C-MAPSS data set summary.

Data set	Training trajectories	Testing trajectories	Operating conditions	Fault modes
FD001	100	100	1	1
FD002	260	259	6	2
FD003	100	100	1	1
FD004	248	249	6	2

C-MAPSS: Commercial Modular Aero Propulsion System Simulation.

$R_{early} = 125$ is assumed for all data sets so to be able to compare the results from the proposed model with the ones from other relevant works on C-MAPSS.[31,32,45–47]

As discussed in section "Proposed CapsNets for RUL estimation," a time window of size $N_t \times N_s$ is used to train and validate the proposed model. To build these time windows, the following procedure is used:

1. Clean the data set, dropping sensor measurements that are constant in time, thus leaving 14 sensors to train the model (i.e. $N_s = 14$).
2. Use the shortest test sequence as the length of the window for each sub-data set. Then, $N_t = 30$ for FD001, $N_t = 21$ for FD002, $N_t = 30$ for FD003 and $N_t = 19$ for FD004.
3. Apply Min-Max normalization between $[0, 1]$ using equation (5) for each sensor measurement in the whole training set before producing the time windows of size $N_s \times N_t$, thus avoiding overfitting over measurements with high values. This fitted normalization (over only the training set) is then applied to the test set

$$x_{normalized} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{5}$$

Data augmentation is also used to train the model. To do this, each time window is generated with an overlap of $N_t - 1$ data points in the training set, increasing the available data to train the model. It is important to

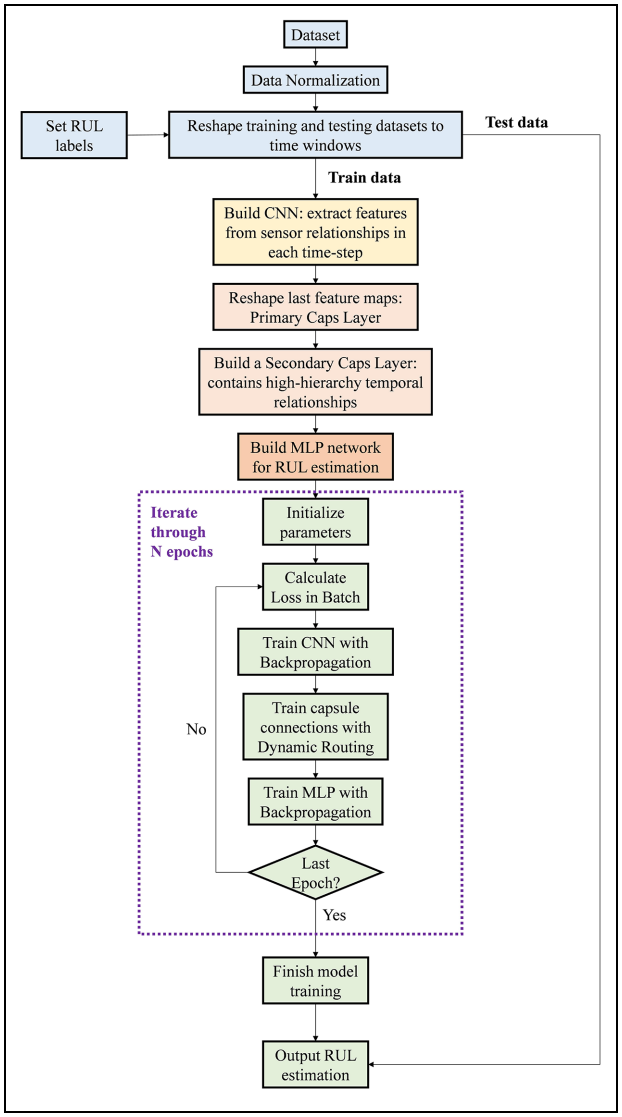

Figure 7. Flowchart for RUL estimation via the proposed CapsNet.

remark that the time windows of normalized raw sensor data are used directly as an input for training, thus not requiring any prior expertise in signal processing.

Model training

To train the model, each training set was randomly split into training and validation sets at a ratio of 85% and 15%, respectively. The validation set was used to measure the model performance in each training epoch. Then, for the training set, the flowchart shown in Figure 7 is employed with RMSprop algorithm[48] for convergence optimization, using a total of 120 training epochs and a batch size of 150 time windows. Moreover, a varying learning rate (l_{rate}) is adopted as proposed by Li et al.,[32] using $l_{rate} = 0.001$ for the first 70 epochs and then changed to $l_{rate} = 0.0001$ for the remaining epochs. This ensures a faster optimization at the beginning and, later, a more stable convergence. In addition, initializer proposed in Glorot and Bengio[49] is used for weight initializations. To prevent overfitting, L2 regularization[50] is used in the convolutional layers, and a dropout regularization rate of 0.2 is applied to both fully connected MLPs (i.e. 20% of input units would be dropped for each batch). The model was developed using Python v3.6 and TensorFlow v1.7 library and trained in a PC with Intel Core i7 6700K CPU, 32 GB DDR4 RAM and 12 GB NVIDIA Titan XP GPU.

Results and sensitivity analysis

To evaluate the performance on RUL estimation in the C-MAPSS data sets, two metrics are used: RMSE and scoring function. RMSE is shown in equation (4) and calculated through every training sample in each sub-data set. The scoring function given below (equation (6)) was proposed by Saxena et al.[15] with the purpose to penalize late RUL predictions over early ones

$$s = \begin{cases} \sum_{i=1}^{N} \exp\left(-\frac{d}{10}\right) - 1 \text{ for } d < 0 \\ \sum_{i=1}^{N} \exp\left(\frac{d}{13}\right) - 1 \text{ for } d \geqslant 0 \end{cases} \quad (6)$$

where N is the number of test samples and d is the difference between the predicted RUL and the targeted true RUL (i.e. $d = y_p - y_t$).

The proposed CapsNets for RUL prognostics are demonstrated by comparing it with the latest state-of-the-art results for each test dataset of the C-MAPSS data set. To do this, every obtained result was averaged through 10 runs. Also, a sensitivity analysis of the model performance is presented for its main components: primary and secondary capsules layer and the number of rounds of dynamic routing between them.

RUL prognostics performance. The RUL prognostics performance of all turbofans testing samples from data sets FD001, FD002, FD003 and FD004 are presented in Figure 8, sorted from the lowest to highest RUL.

As shown in Table 1, data sets FD002 and FD004 were obtained under six different operational conditions (instead of one as in FD001 and FD003). Also, the testing and training trajectories in these data sets were almost 2.5 times more than in data sets FD001 and FD003, showing that FD002 and FD004 data sets are considerably more heterogeneous and complex in nature. This can be clearly seen in Figure 8, where higher noise is observed in the RUL predicted values of the proposed CapsNet model for these data sets, although the behavior of the targeted RUL function is well estimated. Also, it can be observed that the predicted RUL values are generally close to the actual values for the simpler FD001 and FD003 data sets,

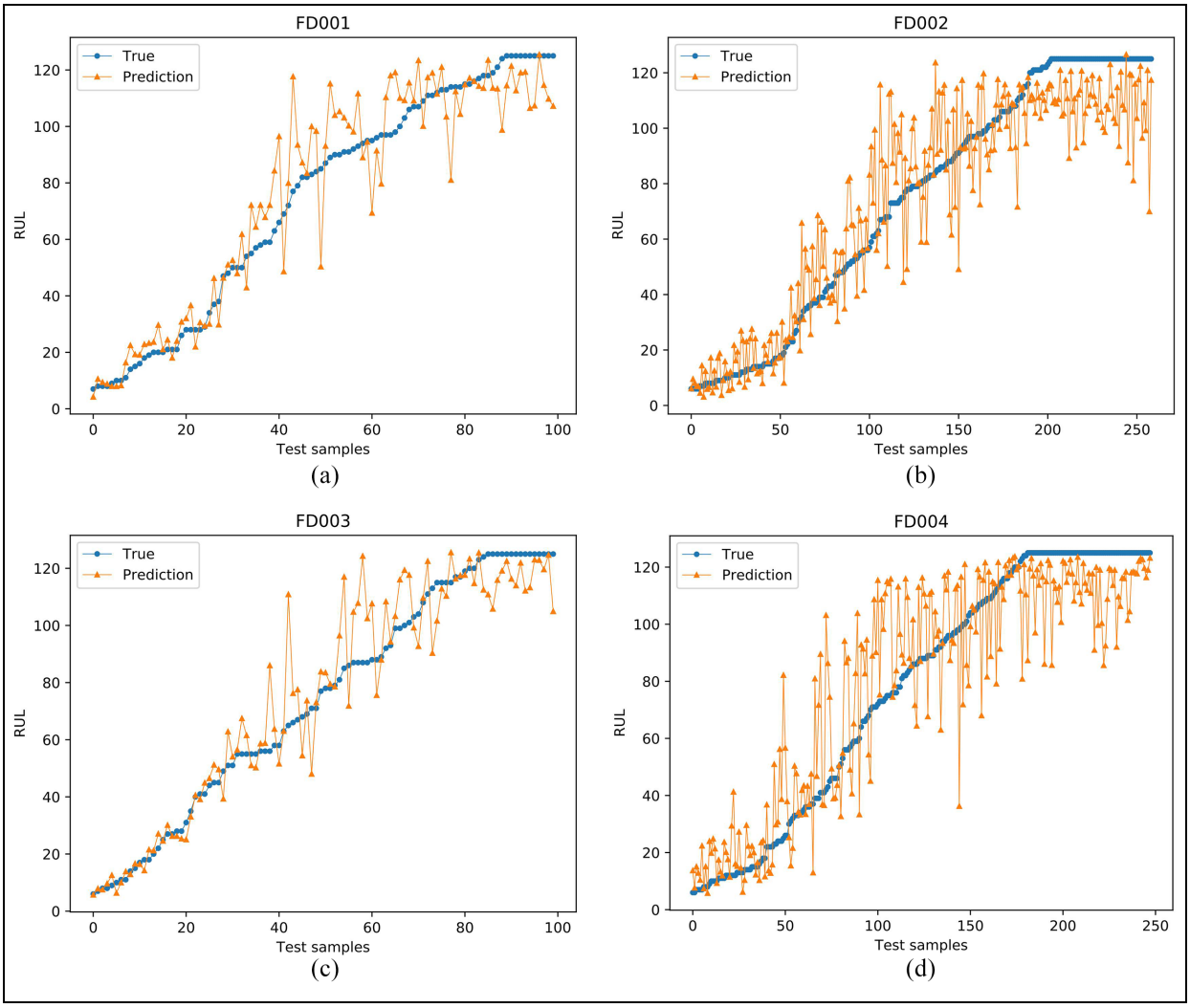

Figure 8. Predicted RUL for each test engine sample in each data set: (a) FD001, (b) FD002, (c) FD003 and (d) FD004.

showing that for only one operation condition, the proposed model prognostics performance is rather insensitive to the number of faults modes presented by the test-engines units. Therefore, it is inferred that the proposed model's performance is more sensible to changes in the number of operational conditions than in the number of fault modes.

Figure 8 also shows that for smaller RUL values, the prediction accuracy increases for all data sets. When the engine is closer to the failure threshold, features attributed to the apparition of faults are enhanced in the proposed model feature extraction, increasing its prognostics performance. This can be seen in Figure 9 where a single engine sample in each data set is used to show its whole lifetime RUL evolution. It should be noted that these particular samples were selected due to its long lifetimes in comparison with the whole data set. Figure 9 shows that, initially, the rectified RUL zone is slightly underestimated by the model. Afterward, the linearly decreasing targeted behavior is well predicted by the model for each data set. Also, for the late period of the engine lifetime, the prognostics performance accuracy increases, which is very critical for health management in industrial applications, leading to an enhanced operation performance and safety.

Furthermore, with the aim of showing the stability of the prediction accuracy regarding its proximity to failure, Figure 9 shows the relative error, estimated according to equation (7), associated with each data set sample penalizing mispredictions nearer to the sample end life

$$e_r = \frac{|y_p - y_t|}{y_t} \tag{7}$$

where e_r is the relative error, y_p is the predicted RUL and y_t is the ground truth RUL of the test sample. Based on Figure 9, e_r is small for each data set, but, as expected, larger for FD002 and FD004 test samples compared with FD001 and FD003. Note that, in general, e_r has a noisy behavior with respect of test samples time cycles, with similar maximum and minimum values for both long- and short-term RUL estimations,

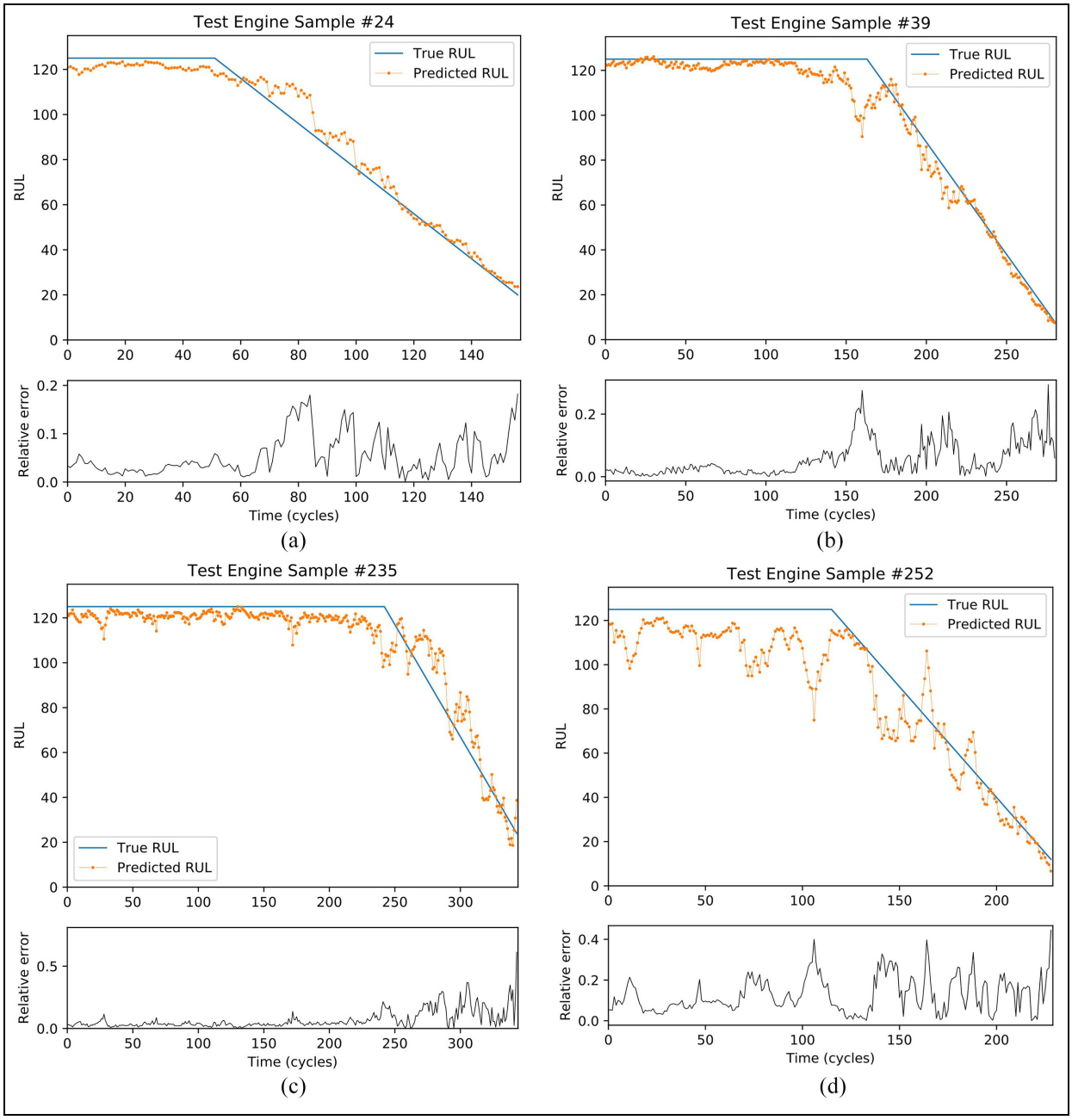

Figure 9. Predicted lifetime RUL and its associated relative error for a sample test engine sample in each data set: (a) FD001, (b) FD002, (c) FD003 and (d) FD004.

showing that the prediction capabilities do not have a direct dependency to the test samples' operational time cycles.

The performance of the proposed CapsNet model for RUL estimation is compared in Table 2 with other relevant models on the C-MAPSS data sets. In particular, we compare against DLSTM model proposed by Zheng et al.,[13] the MODBNE proposed by Zhang et al.,[14] the more traditional CNN model with pooling layers proposed by Sateesh Babu et al.[31] (which was the first attempt of using a CNN in this data set) and Li et al.[32] deep convolutional neural network (DCNN), which, for the best of the authors' knowledge, has the best average results in these data sets. It is important to notice that all four works make use of an $R_{early} \in [120, 130]$. Table 2 shows that the proposed CapsNet model outperforms the other models for RUL estimation on each data set for both mean RMSE and mean Score, the only exception being the FD001, where Li's model obtained three less points in its Score. Thus, one can argue that the CapsNet is a competent model for RUL prediction under different combinations of operating conditions and fault modes for the turbofans. Also, the improved Scores show that the RUL predictions are more conservative, which is a desirable property in the context of safety and reliability, where early

Table 2. Performance comparison of relevant works on the C-MAPSS data sets.

		DLSTM[13]		MODBNE[14]		CNN first attempt[31]		DCNN[32]		Proposed CapsNet	
		RMSE	Score	RMSE	Score	RMSE	Score	RMSE	Score	RMSE	Score
FD001	Mean	16.14	338	15.04	334.23	18.45	1286.70	12.61	273.70	12.58	276.34
	STD	–	–	–	–	–	–	0.19	24.1	0.25	25.95
FD002	Mean	24.49	4450	25.05	5585.34	30.29	13,570	22.36	10,412	16.30	1229.72
	STD	–	–	–	–	–	–	0.32	544	0.23	53.07
FD003	Mean	16.18	852	12.51	421.91	19.82	1596.20	12.64	284.10	11.71	283.81
	STD	–	–	–	–	–	–	0.14	26.5	0.26	29.46
FD004	Mean	28.17	5550	28.66	6557.62	29.16	7886.40	23.31	12,466	18.96	2625.64
	STD	–	–	–	–	–	–	0.39	853	0.27	266.83

C-MAPSS: Commercial Modular Aero Propulsion System Simulation; DLSTM: deep long-short term memory recurrent neural network; MODBNE: multi-objective deep belief network; CNN: convolutional neural networks; DCNN: deep convolutional neural networks; RMSE: root mean squared error; STD: standard deviation.

Table 3. Relative percentage difference for C-MAPSS sub-data sets between CNN-based models and the proposed CapsNet based model for RUL estimation.

	Sateesh's CNN[31]	Li's DCNN[32]	Proposed CapsNet
RPD (%)	42.28	57.59	36.84

C-MAPSS: Commercial Modular Aero Propulsion System Simulation; CNN: convolutional neural networks; DCNN: deep convolutional neural networks; RUL: remaining useful life; RPD: relative percentage difference.

predictions are preferred over late ones. Note also that only Li's model is explicitly evaluated over 10 runs, thus providing the corresponding RMSE and Scores mean and standard deviation (STD).

Even though, at first glance, the first convolutional filter used in the proposed CapsNet model seems quite dependent of the input data order, it is just used as a dimensionality reduction, which is then summarized by the second convolutional operation. To show the robustness of the model regarding input sensor order, the input data time-windows were shuffled by the columns (i.e. the sensor domain) for 10 runs of the model for FD001 and FD004 data sets. For FD001, the RMSE and Score were 12.59 and 263.88, respectively, showing almost no difference with the presented results in Table 2. The same behavior was presented on FD004, where the RMSE and Score were 18.91 and 2792.86, respectively.

The proposed CapsNet model also performs particularly well for RUL estimation from the more heterogeneous and complex FD002 and FD004 data sets in C-MAPSS. Sateesh Babu et al.[31] presented the first attempt of using a CNN model in this data set, employing a traditional approach with convolutional and pooling layers. Based on the results reported in Table 2, Sateesh's model adapts well for the more simple and homogeneous data in FD001 and FD003 data sets, but for FD002 and FD004, relatively worst performances were obtained. For the DCNN model proposed by Li et al.,[32] the pooling layers were eliminated, achieving better RUL prognostics performance in general, but at the cost of a higher relative percentage difference (RPD) between more homogeneous FD001 and FD003 RMSE compared with the more heterogeneous FD02 and FD004 RMSE. To show this, the RPD between these data sets is obtained using equation (7) and shown in Table 3, where

$$RPD = 100 \times \frac{\left|\overline{\mathrm{RMSE}}_{1-3} - \overline{\mathrm{RMSE}}_{2-4}\right|}{\left(\overline{\mathrm{RMSE}}_{1-3} + \overline{\mathrm{RMSE}}_{2-4}\right)/2} \tag{8}$$

with

$$\begin{aligned} \overline{\mathrm{RMSE}}_{1-3} &= \left(\mathrm{RMSE}_{FD001} + \mathrm{RMSE}_{FD003}\right)/2 \\ \overline{\mathrm{RMSE}}_{2-4} &= \left(\mathrm{RMSE}_{FD002} + \mathrm{RMSE}_{FD004}\right)/2 \end{aligned} \tag{9}$$

Table 3 indicates that a traditional CNN approach with pooling layers shows better generalization on C-MAPSS, being more insensitive to significant changes between data sets. However, the exclusion of pooling layers in a CNN architecture eliminates the only component in it that contributes to invariance of information, which can lead to overfitting as seen in Li's model for the more homogeneous data in FD001 and FD003, and which, in turn, results in a relatively worst prognostics performance for the heterogeneous data sets (FD002 and FD004). This is observable in Table 3, where a higher RPD of 57.59% is obtained by Li's model compared with the 42.28% of Sateesh's. However, it can be observed that the proposed CapsNet model is more insensitive to changes in the data sets due to the use of capsules that aims for equivariance of information in the automatic feature extraction process, thus achieving a much smaller difference in RUL estimation performance between homogeneous

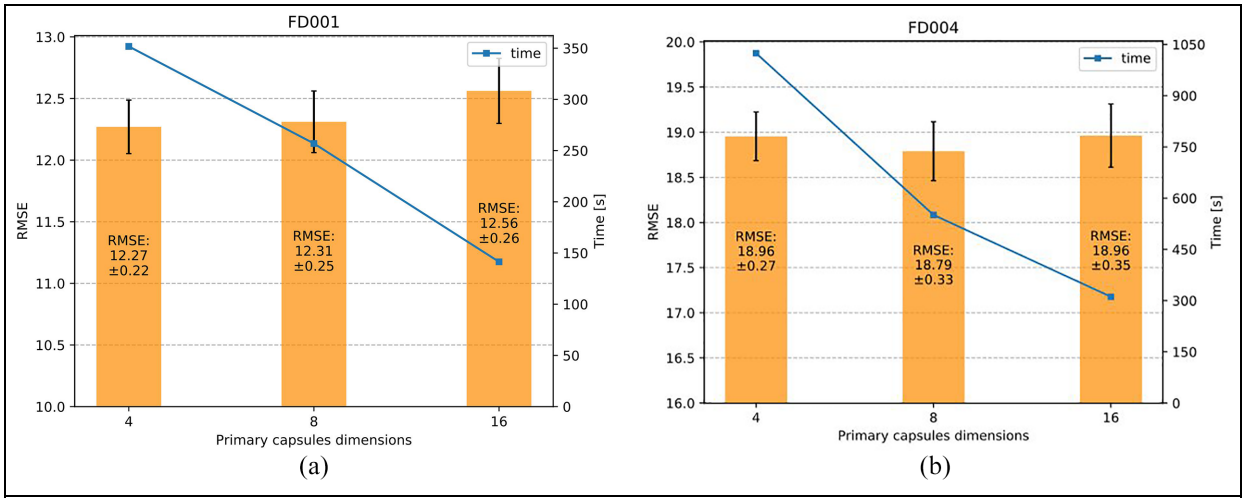

Figure 10. Effect of the dimensions of the primary capsules in the computation time and prognostics performance in the training process of data sets: (a) FD001 and (b) FD004.

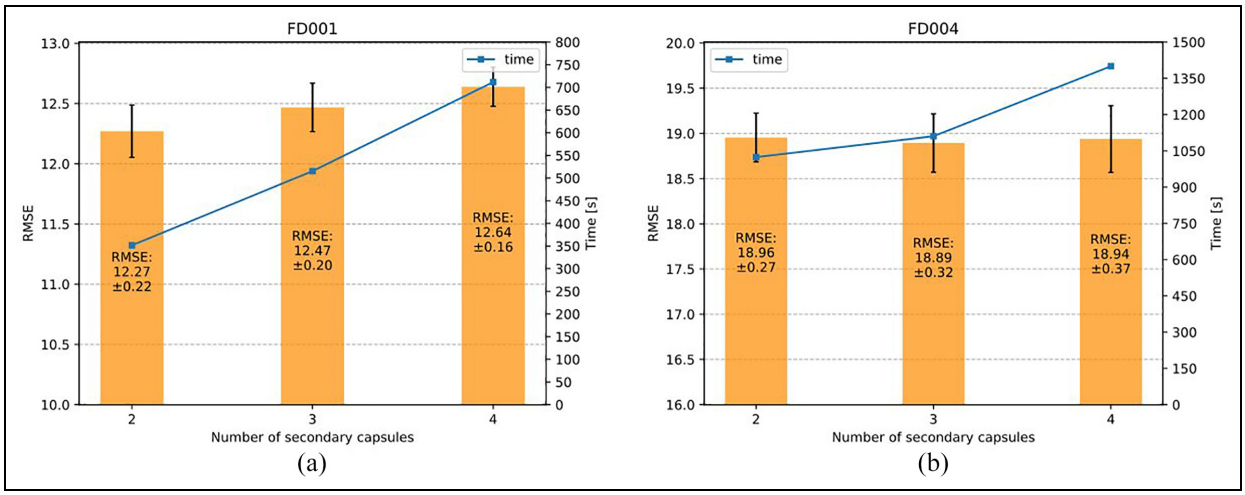

Figure 11. Effect of the number of secondary capsules in the computation time and prognostics performance in the training process of data sets: (a) FD001 and (b) FD004.

and heterogeneous data sets as shown by smaller RPD of 36.84% compared with the other two models.

Sensitivity analysis. In this section, a sensitivity analysis involving the hyperparameters of the proposed CapsNet model for RUL estimation is presented for data sets FD001 and FD004, that is, the simplest and the most complex data sets, respectively. Three hyperparameters of the proposed CapsNet are varied during the training process: the dimensions of primary capsules, the number of secondary capsules and the number of rounds of dynamic routing between them. These hyperparameters were selected because they were the most relevant in determining the quality of the results by the proposed model in terms of RUL performance and Score for all four data sets. Note also that the remaining model's hyperparameters presented in section "Proposed CapsNets for RUL estimation" were left unchanged. Also, to show the robustness of the model with respect to the amount of training data, three different train–validation splits will be analyzed.

Figure 10 shows that an increasing number of dimensions of the primary capsules mostly affect the computation time of the model for both data sets. Fewer dimensions correspond to a greater number of capsules since the feature maps generated by the previous convolutional layer are only reshaped into capsules, thus increasing the number of parameters in the model, which in turn leads to increased training time. However, the RUL prognostic performance is stable both in mean and STD.

Similar behavior is observed in Figure 11 for variations in the number of secondary capsules, with more

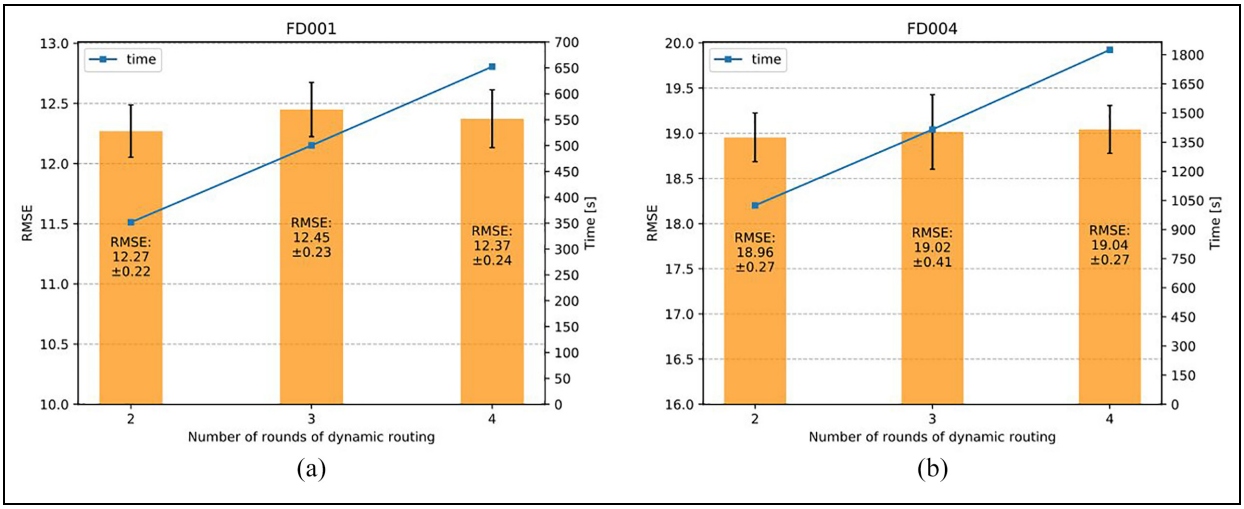

Figure 12. Effect of the number of dynamic routing rounds in the computation time and prognostics performance in the training process of data sets: (a) FD001 and (b) FD004.

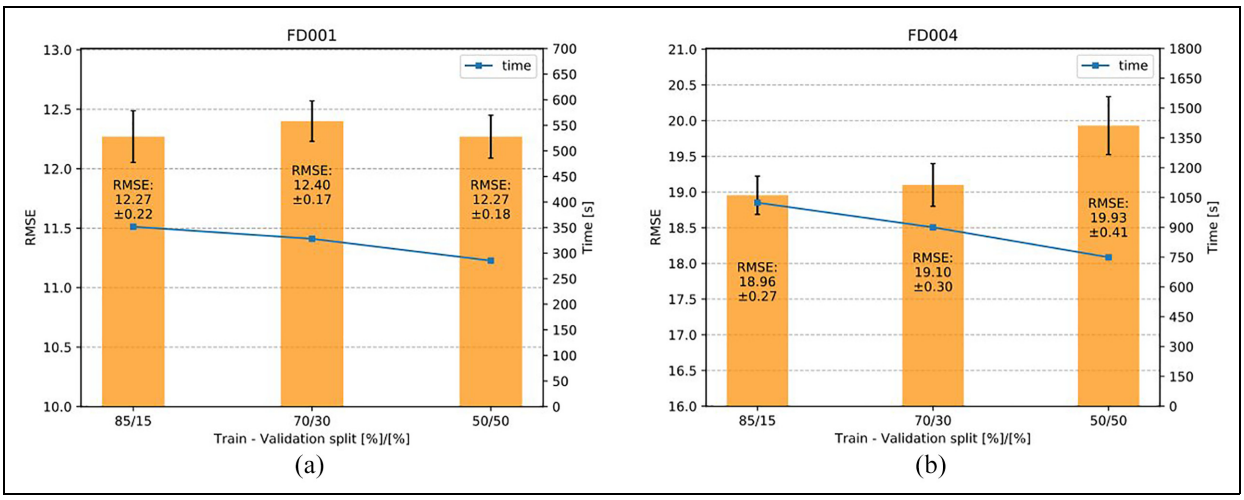

Figure 13. Effect of the number of used training data in the computation time and prognostics performance in the training process of data sets: (a) FD001 and (b) FD004.

capsules leading to an increase in the computation time with no relevant differences in the prognostic performance. Therefore, one can argue that the proposed CapsNet model for RUL estimation seems to be robust in relation to changes in the hyperparameters related to the number and dimensions of capsules.

Another important hyperparameter is the number of rounds of dynamic routing between capsules, since this determines the relationships between low-level features of the data set and high-level ones. Figure 12 shows that the proposed model's prognostic performance does not present relevant differences with an increasing number of rounds for both FD001 and FD004 data sets. However, it can be observed that the training computation time increases linearly with the number of dynamic routing rounds, thus suggesting that no more than two rounds are needed for the proposed model for RUL estimation on C-MAPSS.

Finally, Figure 13 shows the robustness of the model with respect of the amount of available training data. For the simpler FD001 data set, the training data reduced from 85% to 70% and 50% do not show major changes regarding RMSE and training time, showing that the model behavior for more homogeneous data is stable. In the case of the more complex FD004 data set, a slight increase in both RMSE and its STD can be observed, especially for the 50% case, which is expected in the training of a more heterogeneous data set, where less data imply less information about the heterogeneity of the data. Nevertheless, the

proposed model still outperforms other relevant works shown in Table 2, even though almost only half of the available data are used to train the model.

Concluding remarks

Among the different deep learning techniques, CNNs have shown remarkable results on different PHM-related tasks, such as fault diagnostics and RUL prognostics. Despite of this, as discussed in section "CNNs and their limitations," they do have some limitations: translational invariance is only assured using pooling layers that require sub-sampling of its input data, leading to a possible loss of relevant information from the system's sensor data. These limitations can be tackled via CapsNets, which use vector-shaped capsules of neurons instead of single scalar neurons that aim at equivariance of information, ensuring better relationships between low- and high-level features extracted by the network without the need for sub-sampling, so no positional information is lost in the process. Although CapsNets were initially proposed for CV classification tasks,[33,35] the above-mentioned drawbacks of CNNs motivated the proposed model, architecture and algorithm that further develop and explore the flexibility of CapsNets in the context of RUL estimation. The benchmark C-MAPSS turbofans data sets were used as a case study, with state-of-the-art RUL results in both RMSE and Score metrics achieved by the proposed CapsNet model. These results show that the proposed CapsNet model is a competent and promising tool for RUL prognostics and worth further exploring its capabilities in other PHM applications beyond monitoring signals of turbofans.

Indeed, as shown in section "RUL prognostics performance," the proposed CapsNet model for RUL estimation achieved good prognostic performance on the four FD001, FD002, FD003 and FD004 testing sets that conforms C-MAPSS, especially for the late period of the turbofans lifetime. Furthermore, the proposed model showed to be more sensible to the number of operational conditions than the number of fault modes presented by the tested engines, thus achieving better results in the simpler FD001 and FD003 data sets (both with just one operational condition) than in the more complex FD002 and FD004 (both with six operational conditions).

The proposed CapsNet model's prognostics performance also compared favorably with both RMSE and Score metrics obtained from other deep learning–based models. The proposed model showed significant improvements in RUL prognostics on the more complex (i.e. with higher level on heterogeneity) FD002 and FD004 data sets. Also, as presented in Table 3, the use of capsules decreased the relative RMSE difference between simple and complex data sets in comparison with other CNN-based models, a feature that is believed to be triggered by the incorporation of equivariance through the CapsNet.

In addition, as discussed in section "Sensitivity analysis," the proposed CapsNet model showed no relevant sensibility in terms of RUL prognostics when changes are made in its main architecture's components. Therefore, the proposed CapsNet model for RUL estimation seems to be a propitious and robust model for prognostics on PHM-related tasks.

However, the proposed CapsNets are still a new deep learning architecture ought to be further explored. For instance, a not so deep CapsNet model was presented here, leaving for future attempts the analysis of deeper architectures, considering that CNNs automatic future extraction capabilities are usually improved under this approach.[50] Moreover, new routing algorithms might also be considered such as expectation–maximization (EM)-routing.[43]

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship and/or publication of this article.

ORCID iD

Enrique López Droguett https://orcid.org/0000-0002-0790-8439

References

1. Shin J-H and Jun H-B. On condition based maintenance policy. *J Comput Des Eng* 2015; 2: 119–127.
2. Xu P, Wang Z and Li V. Prognostics and Health Management (PHM) system requirements and validation. In: *2010 prognostics and system health management conference*, Macao, China, 12–14 January 2010, pp.1–4. New York: IEEE.
3. Pecht M and Jie Gu J. Physics-of-failure-based prognostics for electronic products. *Trans Inst Meas Control* 2009; 31(3–4): 309–322.
4. San Martin G, López Droguett E, Meruane V, et al. Deep variational auto-encoders: a promising tool for dimensionality reduction and ball bearing elements fault diagnosis. *Struct Heal Monit* 2019; 18(4): 1092–1128.
5. Heng A, Zhang S, Tan ACC, et al. Rotating machinery prognostics: state of the art, challenges and opportunities. *Mech Syst Signal Process* 2009; 23(3): 724–739.
6. Das M, Moura C, Zio E, et al. Failure and reliability prediction by support vector machines regression of time series data. *Reliab Eng Syst Saf* 2011; 96(11): 1527–1534.
7. Gebraeel N, Lawley M, Liu R, et al. Residual life predictions from vibration-based degradation signals: a neural network approach. *IEEE Trans Ind Electron* 2004; 51(3): 694–700.

8. Khan S and Yairi T. A review on the application of deep learning in system health management. *Mech Syst Signal Process* 2018; 107: 241–265.
9. Tian Z. An artificial neural network method for remaining useful life prediction of equipment subject to condition monitoring. *J Intell Manuf* 2012; 23(2): 227–237.
10. Fink O, Zio E and Weidmann U. Predicting component reliability and level of degradation with complex-valued neural networks. *Reliab Eng Syst Saf* 2014; 121: 198–206.
11. Ren L, Cui J, Sun Y, et al. Multi-bearing remaining useful life collaborative prediction: a deep learning approach. *J Manuf Syst* 2017; 43: 248–256.
12. Nectoux P, Gouriveau R, Medjaher K, et al. PRONOSTIA: an experimental platform for bearings accelerated degradation tests. In: *IEEE international conference on prognostics and health management*, Denver, CO, 1–20 July 2012, pp.1–8. New York: IEEE.
13. Zheng S, Ristovski K, Farahat A, et al. Long short-term memory network for remaining useful life estimation. In: *2017 IEEE international conference on prognostics and health management (ICPHM)*, Dallas, TX, 19–21 June 2017, pp.88–95. New York: IEEE.
14. Zhang C, Lim P, Qin AK, et al. Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. *IEEE Trans Neural Networks Learn Syst* 2017; 28(10): 2306–2318.
15. Saxena A, Goebel K, Simon D, et al. Damage propagation modeling for aircraft engine run-to-failure simulation. In: *2008 international conference on prognostics and health management*, Denver, CO, 6–9 October 2008, pp.1–9. New York: IEEE.
16. Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions. In: *Proceedings of the IEEE computer society conference on computer vision and pattern recognition (CVPR)*, Boston, MA, 7–12 June 2015. New York: IEEE.
17. Tompson J, Goroshin R, Jain A, et al. Efficient object localization using convolutional networks. In: *2015 IEEE conference on computer vision and pattern recognition (CVPR)*, Boston, MA, 7–12 June 2015. New York: IEEE.
18. Taigman Y, Yang M, Ranzato M, et al. DeepFace: closing the gap to human-level performance in face verification. In: *Proceedings of the IEEE computer society conference on computer vision and pattern recognition*, Columbus, OH, 23–28 June 2014. New York: IEEE.
19. Sermanet P, Eigen D, Zhang X, et al. OverFeat: integrated recognition, localization and detection using convolutional networks, 2013, https://arxiv.org/pdf/1312.6229.pdf
20. Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE computer society conference on computer vision and pattern recognition*, Columbus, OH, 23–28 June 2014. New York: IEEE.
21. Simonyan K and Zisserman A. Very deep convolutional networks for large-scale image recognition. In: *3rd international conference on learning representations (ICLR 2015)*, 2015, https://arxiv.org/pdf/1409.1556.pdf
22. Ciregan D, Meier U and Schmidhuber J. Multi-column deep neural networks for image classification. In: *2012 IEEE conference on computer vision and pattern recognition*, Providence, RI, 16–21 June 2012. New York: IEEE.
23. Krizhevsky A, Sutskever I and Hinton GE. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*, pp. 1097–1105.
24. Chen Z, Li C and Sanchez R-V. Gearbox fault identification and classification with convolutional neural networks. *Shock Vib* 2015; 2015: 390134.
25. Wang J, Zhuang J, Duan L, et al. A multi-scale convolution neural network for featureless fault diagnosis. In: *2016 International symposium on flexible automation (ISFA)*, Cleveland, OH, 1–3 August 2016, pp.65–70. New York: IEEE.
26. Guo X, Chen L and Shen C. Hierarchical adaptive deep convolution neural network and its application to bearing fault diagnosis. *Meas J Int Meas Confed* 2016; 93: 490–502.
27. Loparo K. *Bearing data center*. Case Western Reserve University, 2013, http://csegroups.case.edu/bearingdatacenter/pages/welcome-case-western-reserve-university-bearing-data-center-website
28. Verstraete D, Ferrada A, Droguett EL, et al. Deep learning enabled fault diagnosis using time-frequency image analysis of rolling element bearings. *Shock Vib* 2017; 2017: 5067651.
29. Bechhoefer E. A quick introduction to bearing envelope analysis, 2016, https://mfpt.org/fault-data-sets/
30. Modarres C, Astorga N, Droguett EL, et al. Convolutional neural networks for automated damage recognition and damage type identification. *Struct Control Heal Monit* 2018; 25: e2230.
31. Sateesh Babu G, Zhao P and Li X-L. Deep convolutional neural network based regression approach for estimation of remaining useful life, 2016, pp.214–228, https://oar.a-star.edu.sg/jspui/bitstream/123456789/1681/3/DASFAA2016_014_final_v1.pdf
32. Li X, Ding Q and Sun JQ. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliab Eng Syst Saf* 2018; 172: 1–11.
33. Hinton GE, Krizhevsky A and Wang SD. Transforming auto-encoders. In: *Lecture notes in computer science (Including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)*, 2011, https://www.cs.toronto.edu/~hinton/absps/transauto6.pdf
34. Toledo DG, Meruane V, Droguett EL and Moderres M. Acoustic emission based fault diagnosis via a novel deep convolutional neural network method. In: *Safety and Reliability -Safe Societies in a Changing World*, pp. 1157–1163. CRC Press.
35. Sabour S, Frosst N and Hinton GE. Dynamic routing between capsules. In: *Advances in neural information processing systems*, pp. 3856–3866.
36. LeCun Y and Cortes C. MNIST handwritten digit database. *AT&T Labs*, 2010, http://yann.lecun.com/exdb/mnist
37. Zhu Z, Peng G, Chen Y, et al. A convolutional neural network based on a capsule network with strong generalization for bearing fault diagnosis. *Neurocomputing* 2019; 323: 62–75.
38. Lessmeier C, Kimotho JK, Zimmer D and Sextro W. Condition monitoring of bearing damage in

electromechanical drive systems by using motor current signals of electric motors: a benchmark data set for data-driven classification. In: *Proceedings of the European conference of the prognostics and health management society*, pp. 05–08.
39. LeCun Y and Bengio Y. Convolutional networks for images, speech, and time series. In: Arbib MA (ed.) *The handbook of brain theory and neural networks*. Cambridge, MA: MIT Press, 1995, pp.255–258.
40. LeCun Y, Bengio Y and Hinton G. Deep learning. *Nat Methods* 2015; 521: 436–444.
41. Boureau Y-L, Ponce J, Fr JP, et al. A theoretical analysis of feature pooling in visual recognition. In: *Proceedings of the 27th international conference on machine learning*, Haifa, Israel, 21–24 June 2010. Madison, WI: Omnipress.
42. Kondo S, Yoshida T and Ohki K. Mixed functional microarchitectures for orientation selectivity in the mouse primary visual cortex. *Nat Commun* 2016; 7: 13210.
43. Sabour S, Frosst N and Hinton G. Matrix capsules with EM routing. In: *6th international conference on learning representations (ICLR 2018)*, Vancouver, BC, Canada, 30 April–3 May 2018, https://openreview.net/pdf?id=HJWLfGWRb
44. Peel L. Data driven prognostics using a Kalman filter ensemble of neural network models. In: *2008 international conference on prognostics and health management*, Denver, CO, 6–9 October 2008. New York: IEEE.
45. Heimes FO. Recurrent neural networks for remaining useful life estimation. In: *2008 international conference on prognostics and health management*, Denver, CO, 6–9 October 2008. New York: IEEE.
46. Malhotra P, Vishnu TV, Anusha Ramakrishnan, et al. Multi-sensor prognostics using an unsupervised health index based on LSTM encoder-decoder. In: *Presented at 1st ACM SIGKDD workshop on machine learning for prognostics and health management*, San Francisco, CA, August 2016, https://arxiv.org/pdf/1608.06154.pdf
47. Ramasso E. Investigating computational geometry for failure prognostics. *Int J Progn Heal Manag* 2014; 5: 1–18.
48. Dauphin YN, De Vries H and Bengio Y. Equilibrated adaptive learning rates for non-convex optimization. In: *Proceedings of the 28th international conference on neural information processing systems (NIPS'15)*, Montreal, QC, Canada, 7–12 December 2015, pp.1504–1512. Cambridge, MA: MIT Press.
49. Glorot X and Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249–256).
50. Schmidhuber J. Deep learning in neural networks: an overview. *Neural Networks* 2015; 61: 85–117.